A MOBILE, MAP-BASED TASKING INTERFACE FOR HUMAN-ROBOT INTERACTION

By

Eli R. Hooten

Thesis

Submitted to the Faculty of the

Graduate School of Vanderbilt University

in partial fulfillment of the requirements

for the degree of

MASTER OF SCIENCE

in

Electrical Engineering

December 2010

Nashville, Tennessee

Approved:

Professor Julie A. Adams

Professor William H. Robinson

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
DEC 2010		**00-00-2010 to 00-00-2010**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
A Mobile, Map-Based Tasking Interface for Human-Robot Interaction	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Vanderbilt University,Nashville,TN,37240**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**73**	

ACKNOWLEDGMENTS

Throughout this work, I was fortunate to be advised by Dr. Julie Adams and Dr. William Robinson, both of whom graciously volunteered to serve on my thesis committee. Both professors were approachable, understanding, and more than willing to take time out of their busy schedules to accommodate my needs. For these reasons, they were invaluable to my success and I am grateful for their participation and involvement.

This thesis would not have been possible without the hard work of two dedicated individuals within the Human-Machine Teaming Laboratory at Vanderbilt University, Sean Hayes and Sanford Freedman. Both individuals were pivotal in laying the initial groundwork for the map-based tasking interface described in this work. In the case of Mr. Hayes, the reader will find footnotes throughout this work acknowledging his specific contributions. Mr. Hayes also graciously offered his time to discuss the intricacies of the statistical methods used in this thesis. Mr. Freedman continues to lend his assistance to my research to this day, frequently providing a discerning eye that has been invaluable to improving the research presented in this thesis.

This research was partially supported by a contract from the US Marine Corps Systems Command to M2 Technologies, Inc., National Science Foundation Grant IIS-0643100, and by a Department of Defense National Defense Science and Engineering Graduate Fellowship. Additional partial support for underlying system development has been provided by the Office of Naval Research Multidisciplinary University Research Initiative Program award N000140710749.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **ii**

LIST OF TABLES . . . . . **v**

LIST OF FIGURES . . . . . **vi**

LIST OF ABBREVIATIONS . . . . . **viii**

I Introduction **1**

II Background Information **3**

II.1 Interactive Modalities and Multi-Touch 3
II.2 Communicative Modalities 6
II.3 Map-Based Tasking with Mobile Devices 7
II.4 Geospatial User Input Reduction Methods 7
II.4.1 Introduction 7
II.4.2 Precise Selection and Finger-Based Drawing 8
II.4.3 Finger-Based Drawing and Input Reduction Methods 10
II.4.3.1 Linear Methods 11
II.4.3.2 Curve Methods 11
II.4.3.3 Calculating Reduction Error 14
II.4.4 Conclusion 18

III System Description **20**

III.1 The Mobile, Map-Based Tasking Interface 20
III.1.1 Introduction 20
III.1.2 Previous Interface Design 20
III.1.3 Multi-Touch Interface Design 23
III.1.4 The Communications Mode 28
III.1.5 Mobile, Map-Based Tasking Interface Design Summary 31
III.2 Reducing Geospatial User Input 32
III.2.1 Linear Methods for Reducing User Input 33
III.2.2 Curve Methods for Reducing User Input 34
III.2.3 Error Calculation of Reduced Data Sets 36
III.2.4 Reducing User Input Summary 38
III.3 System Description Summary 39

IV Communications Mode Evaluation **40**

IV.1 Introduction 40
IV.2 Method 40
IV.2.1 Results 41
IV.2.1.1 Objective Measures 41
IV.2.1.2 Subjective Measures 42

IV.2.2 Discussion 44

V Reduction Methods Evaluation 46

V.1 Introduction 46
V.2 Method 46
V.3 Test Application Description 47
V.4 User Input Method Results 49
V.4.1 Mouse-Based and Touch-Based Input Comparison 49
V.4.2 Discussion 52
V.4.3 Reduction Methods Results 53
V.4.4 Reduction Methods Discussion 59
V.5 Conclusion 60

VI Concluding Remarks 61

VI.1 Introduction 61
VI.2 The Communications Mode 61
VI.3 Reduction Methods 61
VI.4 Future Work 62

BIBLIOGRAPHY 63

LIST OF TABLES

Table		Page
IV.1	Scenario completion durations using each modality.	42
IV.2	Communicative Modality Likert Scale Ratings.	43
IV.3	NASA TLX subjective workload ratings.	43
V.1	Error scores per participant by shape type and input method. Error scores are the summed average of error rates per shape.	50
V.2	Error scores reported by input method.	50
V.3	Error scores reported by shape type.	51
V.4	Error scores for each of the reduction methods for the Mouse-Linear condition.	54
V.5	Results of the Tukey test for the Mouse-Linear condition. Note that all comparisons except Iterative Cubic - Cubic are significant for the Mouse-Linear condition.	54
V.6	Error scores for each of the reduction methods for the Mouse-Curved condition.	55
V.7	Results of the Tukey test for the Mouse-Curved condition. Note that the Iterative - Cubic, Iterative - Iterative Cubic, and Iterative-Slope comparisons are significant	55
V.8	Error scores for each of the reduction methods for the Touch-Linear condition.	56
V.9	Results of the Tukey test for the Touch-Linear condition. Note that all comparisons against the Iterative method are significant.	56
V.10	Error scores for each of the reduction methods for the Touch-Curved condition.	57
V.11	Results of the Tukey test for the Touch-Curved condition. Note that the Iterative-Cubic, Iterative-Iterative Cubic, and Iterative-Slope comparisons are significant.	57
V.12	Mean sizes of the reduced data sets for each reduction method and the input data set categorized by input method. The Reduction value is the difference in set sizes for the reduction method versus the input set.	58
V.13	Mean sizes of the reduced data sets for each reduction method and the input data set categorized by shape type. The Reduction value is the difference in set sizes for the reduction method versus the input set.	59

LIST OF FIGURES

Figure		Page
II.1	A curved line sampled from user input. The yellow circles represent the sampled waypoints. Note that many of these waypoints may not be necessary to accurately describe the line.	10
II.2	A cubic spline, shown in black, and a tangent vector, shown in blue, to a point on the spline.	13
II.3	An example of sampling error. The black line is the original input line, the blue line is a linear interpolation based on the reduced data set. The red line shows the sampling error between a point in the data set and the interpolated line.	15
III.1	The Traditional Interface displaying three robots performing a Cross Danger Area (CDA) task.	21
III.2	Touch input on a Dell XT2 running the TI.	22
III.3	The Create Task Panel. Note the various pull down menus, text fields, check boxes, and buttons used. These components can make touch-based interaction difficult.	22
III.4	The TI visualizing a Go To X on screen. Note the black connecting lines joining each waypoint (colored yellow) on the screen. Completed tasks are visualized without waypoints.	23
III.5	A visual depiction of the pan (left) and pinch (right-top and right-bottom) Multi-Touch gestures.	24
III.6	The MTI displaying a task that was created using the Drawing Mode.	25
III.7	A Cross Danger Area task shown in (a) Drawing Mode and (b) after the task is committed.	26
III.8	The Comms Mode being displayed with several annotations on the screen. The user can choose to display or hide these annotations.	27
III.9	Task labels as represented in the Communications Mode of the MMBTI.	29
III.10	Stylized task displays as seen in the Communications Mode of the MMBTI.	29
III.11	An example Task Set as displayed in the MMBTI.	31
III.12	The Communications Menu.	32
III.13	Slope (black) and Iterative (orange dashed) reduction applied to an input data set (green). The Slope reduced line is partially obscured by the input data set.	34
III.14	Cubic reduction (blue) and Iterative Cubic reduction (red dashed) reduce the input state set (green).	36
III.15	Shape error (red) between a user's input and a predefined hexagonal shape.	38

V.1 The shapes tested. Shapes (a) through (f) are linear, while shapes (g) through (l) are curved. 48

V.2 The Reduction Methods Evaluator application 49

V.3 Averaged error scores for each participant shown by input type. 51

V.4 Average drawing speed for each participant using mouse-based and touch-based interaction. 52

V.5 Average error scores of the four reduction methods for each input condition. 54

V.6 Average set sizes for each reduction method for both mouse-based and touch-based interaction. 58

LIST OF ABBREVIATIONS

API	Application Programmer Interface
CI	Confidence Interval
GUI	Graphical User Interface
HRI	Human-Robot Interaction
IMD	Intermediate-Sized Device
mHRI	mobile Human-Robot Interaction
MMBTI	mobile, map-based tasking interface
MTI	Multi-Touch Interface
TI	Traditional Interface

CHAPTER I

Introduction

Effective communication can be critical in Human-Robot Interaction (HRI) tasks. Communication is especially important in the case where robots are deployed to assist with disaster relief, emergency response, or military applications that require the reliable and rapid transfer of information. It can also be beneficial, or even required, for a human operator to interact directly with robots in the field. The need to work alongside robots imposes a mobile constraint on the hardware and software that must be used. This work proposes a Communications Mode for a Mobile, Map-Based Tasking Interface (MMBTI) to assist in the transfer of information during mobile Human-Robot Interaction (mHRI) tasks.

User input can be crucial to mHRI tasks. Since the user no longer has access to conventional input methods (e.g., a keyboard and mouse), novel interaction methods that utilize finger-based input can be used. In addition, the appropriate display of information must be carefully considered. Since the user is constrained to a mobile device, information will be displayed using smaller screen sizes than a conventional desktop personal computer. Therefore, this work will investigate methods for reducing geospatial (i.e., coordinate-based) user input in order to display that information compactly and efficiently.

The contribution of this work is two-fold. First, a Communications Mode has been developed in order to assist with mHRI. The Communcations Mode attempts to improve usability through a novel presentation of task-specific information within a map-based interface. In order to determine the usefulness of the Communications Mode, it was subjectively and objectively compared to two other methods of conveying task-specific information to a mobile user. These two additional methods were paper-based tasking, which provides information transfer via an annotated paper map, and auditory-tasking, which transfers information through auditory channels.

The second contribution of this thesis is the development of methods to intelligently reduce geospatial user input. Since the user is mobile, novel input methods (such as a finger interacting directly with a touch screen) must be used to interact with the interface. Therefore, it is important to gain some perspective on the limitations of these input methods and provide a means to improve them, if necessary. Through a user study, this work compares input error differences between finger-based input and mouse-based input in order to determine if finger-based user interaction is deficient when compared to mouse-based interaction. This work also develops and tests four methods of reducing user geospatial input in order to represent it as efficiently as possible on a mobile, map-based interface.

One of the most widely utilized finger-based interaction methods is Multi-Touch. Multi-Touch devices

such as Apple's iPhone and iPad have seen widespread commercial success with combined sales of over 85 million units (Kincaid, 2010). Multi-Touch research has been conducted in a number of domains, including geospatial navigation, command and control, and education (Schöning, 2010). Numerous studies have also been conducted to evaluate the applicability of Multi-Touch for HRI (Hayes et al., 2010; Micire et al., 2009b; Kato et al., 2009; Micire et al., 2009a). However, much of the Multi-Touch interaction research in the HRI domain has focused on applications where users must remain stationary. Comparatively, very little research attention has been paid to a mobile user (e.g., a user interacting with a computing device while moving through and interacting with the environment).

The appropriate background information for this work will first be provided in the form of a literature review. Following the literature review, the developed MMBTI will be fully described in Chapter III. This work will conclude with two user studies (see Chapters IV and V). The first will present findings from a user evaluation of the Communications Mode. The second will present findings from a user evaluation of the geospatial input reduction methods.

Findings from the Communications Mode evaluation determined that the Communications Mode is superior to tasking using auditory cues. The Communications Mode was also shown to be as good as, if not superior to, paper-based tasking methods. Results from the user input reduction methods concluded that touch-based input has a higher rate of error than mouse-based input, but is a faster form of interaction than mouse-based interaction. A user study of the four methods of reducing user input determined that one particular reduction method, Iterative Cubic reduction, seemed to be the most appropriate for reducing touch-based geospatial input.

CHAPTER II

Background Information

II.1 Interactive Modalities and Multi-Touch

Multi-Touch is a method of touch input that allows multiple fingers to be used simultaneously to interact with a device or interface. While the term Multi-Touch has recently become somewhat of a buzzword, the technology has existed since the 1980s (Westerman, 1999). Recently, Multi-Touch has enjoyed increased popularity due to its incorporation into large touch tables like the Microsoft Surface table (Microsoft, 2010) and small mobile devices such as Apple's iPhone (Apple, 2010b). The use of Multi-Touch in such contrasting form factors (i.e., the large Surface table compared to the diminutive iPhone) has helped to cement it as the popular interaction choice for touch screens of any size.

From the software perspective, Multi-Touch has also seen an increase in popularity. Microsoft© Windows 7 was specifically developed with Multi-Touch interaction in mind (Kiriaty, 2009). New application programmer interfaces (APIs) also exist for Multi-Touch. Nokia's release of Qt 4.6 introduced a new API for Multi-Touch programming (Davies, 2009). Multi-Touch APIs also exist for mobile platforms, including Android (Burnette, 2010).

Multi-Touch's integration into large, tabletop displays and small, mobile devices has prompted interaction research that utilizes both types of devices. As such, much research has been published concerning the use of Multi-Touch on large tabletop displays. Tabletop Multi-Touch research spans a broad array of topics, including interface design techniques (Benko et al., 2006), applicability for single and multi-user environments (Wigdor et al., 2007; Tse et al., 2006), and the navigation and analysis of geospatial data (Forlines and Shen, 2005).

While tabletop Multi-Touch research is useful, it is debatable how applicable these techniques are to mobile Human-Robot Interaction (mHRI). The usefulness of tabletop interaction techniques is questioned for several reasons. First, users interacting with a tabletop interface can typically interact bimanually (i.e., using two hands). This is not true in most mHRI tasks, where a user may be using one hand to physically support the hardware and the other hand to interact with it. Tabletop interaction is also largely constrained to the tabletop. In multi-user settings, individuals may converse with one another, but all tasking typically utilizes the tabletop. An mHRI user may be performing multiple tasks, many of which may not involve the use of a mobile device. The mobile user may have a much higher attentional demand placed on him/her due to the required interaction with the environment. Depending on the deployment environment, the same may not

necessarily be true for a tabletop user. Tabletops also have significantly more screen real estate. Therefore, the interface design does not necessarily display information in the most space efficient form possible. Screen occlusion is also not as large of a concern with tabletop interaction. Since tabletop widgets can be quite large, they have a lower chance of being totally occluded by the user's hands or arms. For all these reasons, tabletop interaction techniques and interface designs are largely inapplicable to mHRI interface design.

Multi-Touch interaction for mobile devices primarily focuses on clever Multi-Touch input methods that attempt to circumvent the issue of limited screen real estate and facilitate interaction using one hand. Examples of this type of research include ThumbSpace (Karlson and Bederson, 2007), MicroRolls (Roudaut et al., 2009), and Bezel Swipe (Roth and Turner, 2009). ThumbSpace and MicroRolls attempt to gain more input flexibility over other unimanual Multi-Touch interaction methods by leveraging thumb-based interaction. Bezel Swipe attempts to utilize differing swipe techniques to allow for varied interaction methods. All three techniques were developed to solve a key problem with mobile Multi-Touch interaction: gaining more input flexibility without increasing input difficulty.

Many human-computer interaction studies concerning modal preference focus on interactive modalities (Cohen et al., 2000, 1998; Oviat et al., 2004). Studies focusing on multimodal input (such as the multimodal pen/voice approach (Cohen et al., 2000)), have shown significant improvements in speed and overall usability versus traditional graphical user interface-based (GUI) interaction. Hayes et al. have shown that, for map-based tasks, Multi-Touch interaction results in preferred and superior performance versus single-touch interaction (Hayes et al., 2010).

mHRI specific applications are severely limited by several factors. These factors include: the mobile device's small screen size (Landay et al., 1993), unimanual (i.e., single handed) operation (Forlines et al., 2007), and the presence of pre-existing gestures that are considered to be ubiquitous. The introduction and general acceptance of gestures such as pinching for zooming in and out, two finger panning for scrolling the interface, and single-finger usage for selection has helped supply a general set of interactions applicable to almost any Multi-Touch enabled interface. Now that these gestures are ubiquitous, they must be considered during an interface's design process. Developers of alternate Multi-Touch interaction methods (such as MicroRolls, ThumbSpace, and Bezel Swipe) paid close attention during the design process to create interaction techniques that worked to enhance, not replace, these features (Roth and Turner, 2009; Roudaut et al., 2009; Karlson and Bederson, 2007). For example, in the case of Bezel Swipe, which was designed to facilitate other commonly needed functions, pinching and panning was reserved for zooming and panning the interface (Roth and Turner, 2009). As more Multi-Touch gestures are developed and become ubiquitous, their function will become the standard behavior for the physical gestures they utilize. Therefore, pre-existing gestures can become formidable design constraints.

It is difficult to gauge the applicability of thumb-based interaction methods, such as MicroRolls and ThumbSpace to mHRI. It can be assumed that mHRI tasks will be carried out in complex, dynamic domains. Techniques such as MicroRolls and ThumbSpace were developed for mobile phones and have not been tested on larger devices. Interface design on a device such as a mobile phone, which typically has a screen no larger than 4" and supports a resolution of 240 x 320 sq. pixels or less (Hjerde, 2008), could be extremely difficult in these domains. Also, thumb-based techniques were developed to permit the user to interact with the device using the thumb of the same hand that is holding the mobile device. With a larger device (i.e., a device that requires a user to support the device using a flat palm), thumb-based interaction is impractical, if not impossible.

It is assumed that the appropriate interface design for mHRI tasks requires some sort of intermediate-sized device (IMD). An IMD is a device with a form factor that allows it to be carried, but is larger than a standard mobile phone. As an estimate, the device should possess a screen larger than 6", but smaller than 14", as a larger screen would make the device cumbersome to use in the field. An IMD is required for a few reasons. First, the available screen real estate of mobile phones may simply be too limiting for mHRI. This assumption may be false depending on the mHRI task domain; however, other factors also make the mobile phone impractical. While mobile device computing power is increasing, it is doubtful that any current mobile phone possesses the computational capabilities necessary to visually relay information from multiple robots, the environment, and others in the domain. Battery constraints are also an issue. Mobile phones are designed to have long lasting batteries, as long as they do not see extended periods of heavy use. An mHRI task can potentially require device usage over extended periods of time, thus requiring more battery capacity than is currently available in a mobile phone.

Recently, Multi-Touch has been utilized for devices larger than mobile phones, but still significantly smaller (and more portable) than large tabletop displays. IMD's include tablet-based computers such as the Dell XT2 (Dell, 2010) and Apple's iPad (Apple, 2010a). The popularity of these larger mobile devices has resulted in a large number of tablet-based computing devices becoming commercially available (Gallagher, 2010), helping to establish these devices as technologically relevant. IMDs provide a suitable form factor for carrying out mHRI tasks due to their larger screen size, extended battery capacity, and improved computational performance versus mobile phones. While IMDs typically do not possess the same level of portability as mobile phones, their benefits to mHRI far outweigh this drawback.

This section has provided the basic definition for Multi-Touch and presented existing Multi-Touch research for tabletop displays and mobile devices. While this research is beneficial to Multi-Touch in general, the developed techniques may not be appropriate for mHRI tasks. Both mobile phones and tabletop displays have been shown to be inappropriate for mHRI tasks. However, new IMDs, such as the iPad and tablet

personal computers, appear to be appropriate for mHRI tasks.

II.2 Communicative Modalities

Prior efforts have determined that the use of audio as a communicative modality is not always appropriate for a given task or environment. Audio-based communicative modalities should be avoided for complex content, where visual representations should be used whenever possible (Smith and Mosier, 1986). However, Dowell and Shmueli have shown that audio is not a detriment when redundantly paired with visual displays (Dowell and Shmueli, 2008). Cacciabue and Martinetto demonstrated that audio as a communicative modality can be helpful when used in domains where the user may not always be able to access the information visually, such as in-vehicle collision avoidance systems (Cacciabue and Martinetto, 2006). Cacciabue and Martinetto's in-vehicle collision avoidance system used audio, along with a visual interface to issue warnings if a collision was imminent.

The benefit of using audio to reinforce visual data has been attributed to the sharing of workload across two perceptual channels (Moreno and Mayer, 2002). This idea conforms with multiple-resource theory (Wickens and Hollands, 1999), and delivering redundant information through multiple channels should offload information from one potentially overloaded channel to another. However, other studies have shown that redundant multimodal interfaces can hinder usability. Wickens et al. developed a redundant multimodal display for navigational instructions in a flight simulator that performed worse than any single modality (i.e., a visual only or audio only modality) (Wickens et al., 2003). In many cases, redundant multimodal interfaces appear to be no better, or worse, than their single-modality counterparts (Wickens and Gosney, 2003).

It appears that the benefit of redundant multimodal interfaces is domain and interface dependent. Seagull et al. developed a simulated intensive care system, for which users found its multimodal presentation frustrating (Seagull et al., 2001). This frustration was attributed to the system's reading of text aloud as the user was also reading that same information visually. Mismatches in the narration speed versus the users' reading speed were the source of the frustration. As a counterexample, Cacciabue and Martinetto's system provided redundant information multimodally in a way that was clear, effective, and easy to use (Cacciabue and Martinetto, 2006).

Based on these mixed results, it is worthwhile to investigate various communicative modalities in a mobile setting. It is important to determine which modalities are ideally suited for mobile use. While it is suspected that no ideal modality will be determined, it can be shown which modalities perform better than others. For example, interfaces that verbally convey information to the user may perform better than those that use a purely visual presentation. Furthermore, it can be shown which modalities may hinder the mobile user.

II.3 Map-Based Tasking with Mobile Devices

ł”abelmapBasedTasking Mobile interface design can be extremely challenging. Chittaro points out many challenges that arise in mobile interface design due to constraints placed upon the designer by the mobile device (Chittaro, 2006). These constraints include limited display size, varying aspect ratios and screen resolutions across different devices, hardware inadequacies, form factor, and available input methods. All of these factors must be considered when designing a mobile interface. These constraints can make the design of a map-based interface particularly challenging, since these interfaces require a spatial representation of information that may be space consuming on the screen. Rendering maps can also be computationally expensive depending on the level of map detail, thus hardware constraints must also be considered.

Map use for mobile applications has primarily been concerned with the decisions a user must make and the spatial information required for those decisions. Maps are also used as a compensation tool that can help remedy a user's lack of knowledge when solving spatial problems (Hunolstein and Zipf, 2003). Maps can be especially powerful when they are context-aware and present information accordingly. Nivala and Sarjakoski showed that by embedding maps with context sensitive information, such as the user's location, information concerning nearby items of interest or resources, and time sensitive information, the usability of a map-based user interface can be improved (Nivala and Sarjakoski, 2003). Map-based tasking has been shown to benefit from the use of map-based guides and task visualizations (Hunolstein and Zipf, 2003).

Hunolstein and Zipf's results support the need for map-based guides that support tasking (Hunolstein and Zipf, 2003). They designed a mobile, map-based system for tourists and concluded that mobile, map-based interfaces can provide the same affordances as traditional paper maps, while also providing task-based functionality and task-oriented visualizations.

It appears that map-based tasking can benefit from providing context-sensitive information necessary for task completion. However, caution must be used when attempting to provide information on the map in a mobile setting. Since screen real estate is limited, providing too much information can result in screen clutter and increased cognitive effort. For mobile, map-based tasking, task-critical information must be presented within the device's constraints.

II.4 Geospatial User Input Reduction Methods

II.4.1 Introduction

In the MMBTI, geospatial user input is sampled when the user adds waypoints to an on-screen map in the interface. Waypoints are defined by their x and y-coordinate locations relative to the on-screen map. The methods by which a user can provide geospatial information vary. For example, the user can provide waypoints by clicking and dragging on the map with a mouse. Using this method, waypoints would be added

in a continuous line as the user dragged the mouse across the map. In this example, adding waypoints to the map is analogous to using the mouse to draw a freehand line in a drawing program such as Microsoft© Paint. There are other methods to specify geospatial information to a map-based interface (such as typing the *x*-*y* coordinates of each waypoint or using a single mouse click to specify individual waypoints), but the"drawing" method will be the focus of this review.

The geospatial input data provided by the user forms an input data set, **P**, that is comprised of individual waypoints. When the individual elements of **P** are shown on the on-screen map within the MMBTI and connected by straight lines, a linear spline is formed that represents the user's geospatial input. This section explores methods by which **P** can be reduced and the resulting reduced data set still accurately depict the drawn line that would have been constructed if all of the points in **P** had been used. Two method types for reducing **P** will be discussed: linear methods and curve methods. In addition, the methods for calculating error between the reduced data set and **P** will be discussed.

Before introducing the reduction methods, the topic of drawing on a touch-screen will be discussed. Since the MMBTI relies solely on finger-based input, it is important to discuss the inherent flaws and benefits to touch-based drawing. Techniques for improving the usability and reliability of touch-based interaction will also be discussed in the context of touch-based drawing.

II.4.2 Precise Selection and Finger-Based Drawing

While the drawing method introduced in Chapter II.4.1 was described within the context of mouse-based interaction, the use of a mouse is not necessarily required. In fact, for mobile interaction, where the use of a mouse may be prohibitive, a user's finger can be substituted. The use of a finger as a drawing tool for mobile interaction is not new. In fact, several commercial software programs (such as Adobe's Ideas (Shankland, 2010) application and Autodesk's Sketchbook Pro (Paul, 2010) application for the Apple iPad) for mobile platforms allow the user to create rich, expressive drawings using touch-based interaction. However, current mobile drawing applications are intended for artistic purposes, where drawn lines do not necessarily have to be precise. Map-based interfaces incorporate drawn lines that correspond to collections of waypoints that specify *x* and *y*-coordinates in a physical space. These waypoints may have to be precise for an mHRI task. For example, if a user wanted to instruct a robot to follow a winding road, the user would draw a line along the road as represented on the on-screen map. However, since the finger is not precise (Benko et al., 2006), the user may inadvertently specify the wrong path for the robot, leading to a potential task failure.

Previous research has established that touch-based input is the one of the fastest but least accurate methods of interaction (Muratore, 1987; Ahlström and Lenman, 1987; Karat et al., 1986) . However, these findings have been refuted by Forlines et al. (2007) who determined, for tabletop displays, that touch-based input

may not result in greater speed or accuracy for unimanual tasks versus mouse-based interaction. In a study concerning goal-directed drawing tasks using relative input devices (e.g., mice, trackballs, etc.) and direct input devices (e.g., a touch-screen and stylus), Meyer et al. (1994) concluded that the touch-screen was the worst of the devices in terms of speed and accuracy. Meyer et al. went as far as to conclude that relative mapping is superior to absolute mapping.

The lack of accuracy in touch-based interaction stems from the difficulty of precise selection. Precise selection is inherently difficult in touch-based interaction due to the size of the finger itself. Also, imprecise selection can result as the finger's contact area with the screen changes during finger movement. The changing contact area results in a non-uniform mapping between the fingertip and the point of interaction on the screen, thus resulting in errors (Forlines et al., 2007). Unlike a mouse, which can accurately select a single pixel on the screen, the contact area of the finger on the screen is much larger, resulting in imprecise mappings to a single pixel (Benko et al., 2006). As was indicated in Chapter **??**, many researchers have developed clever methods to alleviate this issue in order to allow touch-based, precise selection. Some precise selection methods have also been developed specifically for precise drawing (Albinsson and Zhai, 2003).

One of the most obvious (and easiest to implement) solutions to precise drawing is to use zooming. If a user of a map-based interface needs to specify a more precisely drawn line, the user can simply zoom the map in very close, decreasing the coordinate-to-pixel ratio and thus allowing the creation of a more accurately drawn line. Zooming has been used for line drawing with positive results (Bederson and Hollun, 1994; Guiard and Beaudouin-lafon, 2004); however, utilizing zooming to ensure precise drawing is not an appropriate solution for mHRI tasks. While specifying tasks for a robot, a user may need to maintain a low zoom level in order to see more of the map and plan out the best route for the robot. Also, tasks that cover large geospatial areas may be difficult to create at a high zoom level. To create a task at a high zoom level, the user may have to draw part of the task, zoom or pan to another area of the map, draw another part of the task, then zoom out to ensure the entire task was correct before committing the task. Therefore, zoom-based approaches for line drawing in mHRI seem cumbersome and less than ideal.

Albinsson and Zhai (2003) proposed a technique called Cross-Lever that allows single-point adjustment of input. Cross-Lever allows a user to specify a point and then draw two perpendicular lines that intersect at the specified point. The user can drag the two lines to adjust the point of intersection, thus adjusting the placement of the specified point. While this method is applicable in the case of single-point adjustment, it is not ideal for drawing, since a user may wish to adjust many points in the drawn line. Albinsson and Zhai also proposed other methods for precise specification of points, such as using on-screen cursor keys to "nudge" individual points and a method utilizing a single handle to push a point into a more precise location (Albinsson and Zhai, 2003). While both of these approaches are applicable to single-point interaction, they

Figure II.1: A curved line sampled from user input. The yellow circles represent the sampled waypoints. Note that many of these waypoints may not be necessary to accurately describe the line.

can become excessively tedious if the user must specify every single point in a drawn line using them.

Potter, Weldon, and Shneiderman proposed a method for continuous (rather than single-point) precise selection with the Take-Off technique (Potter et al., 1988). Take-Off offsets the location of user input to be directly above the user's finger. This offset is denoted by placing a crosshair on the map where the point will be placed if the user draws. Using an offset allows the user to see where the drawn line will be placed as he/she draws, reducing finger-based drawing to a pixel-precise operation. However, Potter et al. reported that this method led to considerably high numbers of selection errors (Potter et al., 1988), which indicates that the Take-Off method may not be suitable for mHRI.

II.4.3 Finger-Based Drawing and Input Reduction Methods

Aside from the difficulties of precise input, a geospatial input method that utilizes finger-based drawing offers other challenges. For example, since the user is supplying input in a continuous fashion, data reduction is limited to the sampling rate of the instrument being used (in this case, the touch screen). If a user provides input at a rate faster than the sampling rate of the instrument, data can be missed.

Problems also arise if the user supplies input very slowly. If a user draws a line much slower than the sampling rate of the device, excessive waypoints will be generated. Excessive waypoints are those that may not be necessary to describe the line that the user is attempting to draw. An example of excessive waypoints can be seen in Figure II.1. These excessive waypoints are unnecessary data and could possibly be removed. Removing the excessive waypoints will result in less memory overhead for storing the waypoints of a drawn line as well as lower the amount of on-screen elements, thus reducing visual clutter.

From a usability standpoint, the problem of drawing faster than the sampling rate of a touch screen is not an issue, since current capacitive touch screen technology operates between 40 Hz and 60 Hz[1]. Due to such a high sampling rate, a user must draw extremely fast to cause data loss, potentially much faster than a user can draw and still maintain any degree of accuracy. However, creating unnecessary waypoints due to oversampling is an issue. Therefore, investigating methods to reduce the waypoints present in a drawn line is a worthwhile pursuit. Also, waypoints cannot be arbitrarily removed from the data set, since this may alter the characteristics of the drawn line and result in a reduced data set that does not match the input data set.

Therefore, more intelligent methods of reducing the data set must be employed.

Methods have been developed that can reduce the input data set and still represent the drawn line with some degree of accuracy. The methods used in this thesis fall into two categories: linear methods and curve methods. Linear methods attempt to reconstruct the drawn line by reducing the error set based on some predefined metric and then connecting the remaining points using straight lines. Curve methods also reduce the input set, but rather than reconstruct the line from the reduced data set using straight lines, Bézier curves are used. Chapters II.4.3.1 and II.4.3.2 will describe the mathematical background behind these methods.

II.4.3.1 Linear Methods

Compared to curve methods, the linear methods used in this thesis are very straightforward. Therefore, only a simple mathematical introduction is necessary. The linear methods employed in this work utilize either a slope-based or iteration-based reduction process. Mathematical slope is defined as the ratio of the change in a linear segment's altitude change compared to the change in its horizontal distance. The slope, m, of a line can be measured between any two collinear points defined as (x_1, y_1) and (x_2, y_2) with Equation II.1 as follows:

$$m = \frac{y_2 - y_1}{x_2 - x_1}. \tag{II.1}$$

Using a calculation of slope between any two points, the necessity of the points can be determined. For example, given set $\mathbf{P} = \{p1, p2, p3\}$ where $p1$, $p2$, and $p3$ are points on a two-dimensional Cartesian plane, the slope between $p1$ and $p2$ and $p2$ and $p3$ can be calculated using Equation II.1. The difference between the two slopes can then be calculated. If this difference is zero, $p1$, $p2$, and $p3$ are collinear and point $p2$ can be eliminated from $\mathbf{P}$ since $p2$ provides no additional information about $\overline{p1p3}$.

Iteration-based reduction attempts to reduce the input data set by removing members from the set at a specified interval. The specified interval is a heuristic that must be specified by the implementation. Once the iteration interval, k, has been determined, iterative approaches simply iterate through the data set and remove points that occur at every kth location in the set.

II.4.3.2 Curve Methods

Curve methods utilize Bézier curves to connect waypoints in a reduced data set. A Bézier curve is a parametrized curve of n degree that can be generalized by a Bernstein-Bézier (Salomon, 2006) polynomial as follows:

[1]This sampling rate is for the Dell Latitude XT2, the tablet PC utilized for the user evaluations seen later in this work. The 40 Hz to 60 Hz range was verified by conversations with technicians at Dell Computer.

$$
\begin{aligned}
B(t) &= \sum_{i=0}^{n} \binom{n}{i} (1-t)^{n-i} t^i \mathbf{P}_i \\
&= (1-t)^n \mathbf{P}_0 + \binom{n}{1} t\mathbf{P}_1 + \cdots \\
\cdots &+ \binom{n}{n-1} (1-t)^{n-1} \mathbf{P}_{n-1} + t^n \mathbf{P}_n, \quad t \in [0,1]
\end{aligned} \tag{II.2}
$$

where $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$ are points in a Cartesian plane and $\binom{n}{i}$ is the coefficient of the x^k term in the polynomial expansion of $(1+x)^n$ (i.e., the binomial coefficient).

While Equation II.2 provides a generalized means of expressing a Bézier curve, it is not necessary for this work, which limits its use of the Bézier curve to the cubic Bézier. The cubic Bézier is given by the following parametric equation (Salomon, 2006):

$$
B(t) = (1-t)^3 \mathbf{P}_0 + 3(1-t)^2 t\mathbf{P}_1 + 3(1-t)t^2 \mathbf{P}_2 + t^3 \mathbf{P}_3, \quad t \in [0,1]. \tag{II.3}
$$

As can be seen in Equation II.3, the cubic Bézier is represented by four points, $\mathbf{P}_0, \mathbf{P}_1, \mathbf{P}_2$,and $\mathbf{P}_3$. Assuming that the start point of the curve is given by $\mathbf{P}_0$ and the endpoint given by $\mathbf{P}_3$, then $\mathbf{P}_1$ and $\mathbf{P}_2$ serve as control points for the Bézier curve. Typically, the parametrized curve will not pass through its control points (in this case, points $\mathbf{P}_1$ and $\mathbf{P}_2$). Instead, the control points will be used to provide directional information to the curve.

Cubic Bézier curves are extremely popular in computer graphics for representing complex curved lines, serving as tweening paths for computer animation, and saturation curves for color information (Glassner, 1993). The popularity of the cubic Bézier arises due to its versatility. In theory, a Bézier curve can pass through any set of points in a two-dimensional space. However, depending on the points, the order of the Bézier curve can be of high-degree, resulting in a curve that, while passing through all points, has many local maxima and minima and appears "wiggly" (Salomon, 2006). Using a Cubic Bézier, sets of points can be interpolated using Equation II.2 iteratively to generate a curve composed of many small cubic Béziers that tightly connect the points in the set. The resulting curve is referred to as a cubic spline, which is defined as a set of 3^{rd} degree polynomials that are smoothly connected at certain points. A cubic spline must be smoothly joined, which means that the function and its first $n-1$ derivatives are continuous at the points where two polynomials meet (Glassner, 1993). Due to the condition that the first derivatives of a cubic splines segment must match, the second derivatives of each segment are the same at the interior points of the spline.

A common method for building a cubic spline from a set of points is known as the cubic spline method

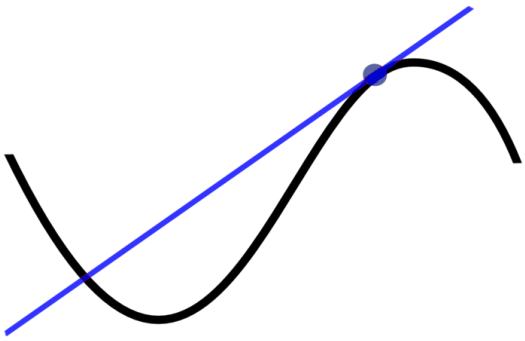


Figure II.2: A cubic spline, shown in black, and a tangent vector, shown in blue, to a point on the spline.

(Salomon, 2006). Given n data points, the algorithm will construct a cubic spline that passes through the points and has the appearance of a single smooth curve. The curve created from the cubic spline method will be composed of $n-1$ individual cubic Bézier curves. Each curve is constructed from overlapping pairs of points in the set of n points, such that points $\mathbf{P}_0$ and $\mathbf{P}_1$ form a pair, $\mathbf{P}_1$ and $\mathbf{P}_2$ form a pair, and $\mathbf{P}_{(n-1)}$ and $\mathbf{P}_n$ form the final pair. So, while points from the input data set form the start and end point of each cubic segment, the control points of each individual segment are unknown.

The cubic spline method is often referred to as an interactive method (Salomon, 2006). The need for the method to be interactive arises due to the unknown control points that must be determined for each individual segment. Fully solving a cubic spline requires solving a system of n equations, where n corresponds to the number of points in the input data set. Each of the n equations represents a tangent vector (see Figure II.2) and each is derived from the requirement that each individual interior segment's second derivative must match those of the other interior segments in the spline. Since a system of n segments is required and the spline only has $n-2$ interior points, the additional two equations must be supplied by alternative means. If the spline is drawn by a user, these two additional equations can be supplied by users in the form of tangent vectors for the 0th and nth point in the data set. Alternative methods also exist to obtain the two additional needed equations. These include treating the spline as a periodic cubic spline, a cyclic cubic spline, or a relaxed cubic spline, among other methods (Knott, 2000).

Regardless of the method used to generate the two equations needed to solve the system of n equations, a general description, obtained from Salomon (2006) of the cubic spline method is as follows:

1. The n data points are input into the program.
2. The additional equations needed of the 0th and nth tangent vector are supplied.
3. The program sets up a system of $n-2$ equations, with the remaining $n-2$ tangent vectors as the unknowns, and solves them.
4. The program loops $n-1$ times. In each iteration it selects two adjacent points and their

tangent vectors to calculate one cubic segment.

A full derivation for obtaining the system of n equations needed to solve for the cubic spline is defined in Salomon (2006). The final system is defined in Equation II.4 as follows

$$\begin{pmatrix} 1 & 4 & 1 & 0 & \cdots & 0 \\ 0 & 1 & 4 & 1 & \cdots & 0 \\ & & & \ddots & \ddots & \vdots \\ 0 & \cdots & \cdots & 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} \mathbf{P}'_0 \\ \mathbf{P}'_1 \\ \vdots \\ \mathbf{P}'_{n-1,n} \end{pmatrix} = \begin{pmatrix} 3(\mathbf{P}_2 - \mathbf{P}_0) \\ 3(\mathbf{P}_3 - \mathbf{P}_1) \\ \vdots \\ 3(\mathbf{P}_n - \mathbf{P}_{n-2}) \end{pmatrix} \tag{II.4}$$

which is a system of $n-2$ equations in n unknowns, $\mathbf{P}'_0$, $\mathbf{P}'_1$, …, $\mathbf{P}'_n$. $\mathbf{P}'_i$ in this case represents the tangent vector for each point i in the set of n points. In II.4 vectors $\mathbf{P}'_0$ and $\mathbf{P}'_n$ are the unknown tangent vectors that must be supplied by the user or some other means. Once the n tangent vectors are determined, they can be used iteratively in Equation II.5 to determine the control points necessary to reconstruct the cubic segment, as described by Equation II.3. Equation II.5 is as follows

$$\begin{aligned} \mathbf{P}(t) = {} & (2\mathbf{P}_{n-1} - 2\mathbf{P}_n + \mathbf{P}'_{n-1} + \mathbf{P}'_n)t^3 \\ + {} & (-3\mathbf{P}_{n-1} + 3\mathbf{P}_n - 2\mathbf{P}'_{n-1} - \mathbf{P}'_n)t^2 \\ + {} & \mathbf{P}'_{n-1}t + \mathbf{P}_{n-1}. \end{aligned} \tag{II.5}$$

II.4.3.3 Calculating Reduction Error

When the data set representing a drawn line is reduced and interpolated, errors between the interpolated line and the original data set can result. For example, when a user draws in the MMBTI, points are sampled from that drawing to form the input data set that is reduced by one of the described reduction methods. Reducing the data set and interpolating the curve formed by the reduced data set can lead to errors. Specifically, these errors are attributed to input points that are not on the line that is created from the reduced data set (see Figure II.3).

Quantifying the reduction error can be generalized to solving the nearest-point-on-curve problem (Glassner, 1993). The problem statement is as follows: given a curve C and some coplanar point P given by (P_x, P_y), find the point on C that minimizes the distance to P. This section introduces the mathematics necessary to solve this problem for the case where C is a cubic Bézier curve and where C is a straight line.

In the case of a linear method being used to reduce an input data set, C is a straight line and the problem is straightforward. When C is linear, it can be described in slope-intercept form as

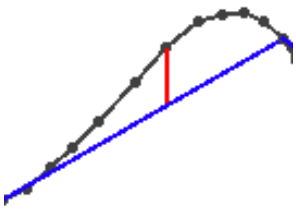


Figure II.3: An example of sampling error. The black line is the original input line, the blue line is a linear interpolation based on the reduced data set. The red line shows the sampling error between a point in the data set and the interpolated line.

$$P_y = mP_x + b. \tag{II.6}$$

The shortest distance between C and P will be formed by a straight line extending from P that intersects C at a right angle at point $C_i = (C_x, C_y)$. Therefore, if C can be described by Equation II.6, then the line from P to a point on C can be described as

$$C_y = -\frac{1}{m}C_x + (P_y + \frac{1}{m}P_x). \tag{II.7}$$

C_i is found by setting Equations II.6 and II.7 equal to one another and solving for C_x to obtain

$$C_x = \frac{mP_y + P_x - mb}{m^2 + 1}. \tag{II.8}$$

Now that C_x is known, Equation II.6 can be used to obtain an expression for C_y

$$C_y = \frac{m^2 P_y + mP_x + b}{m^2 + 1}. \tag{II.9}$$

With C_i known, the distance formula is used to determine the minimum distance, $D_{\overline{pc}}$ from P to the closest point on line C

$$D_{\overline{pc}} = \frac{P_y - mP_x - b}{\sqrt{m^2 + 1}}. \tag{II.10}$$

$D_{\overline{pc}}$ represents the error between the point P in the input data set and the linearly interpolated line C. Each input point will have some error associated with it, even if that error is zero. The error, E, for any given interpolated line is simply the summation of all of the errors associated with the interpolated line

$$E = \sum_{i=0}^{n} D_{\overline{pc}} i. \tag{II.11}$$

Equation II.11 also applies in the case of input sets that are reduced using curve methods.

Deriving the error equation for the curve methods is somewhat more complex than for the linear methods. Consider a curve C that is a cubic Bézier and some point P that is located off of curve C but is coplanar to C. C can be represented by Equation II.3. Since C is a parameterized curve, solving the nearest-point-on-curve problem requires finding the value of t that minimizes the distance from $C(t)$ to P. Just as in the linear case, the shortest distance between P and C will be described by a straight line, $\overline{pc}$. However, unlike the linear case, this $\overline{pc}$ is not guaranteed to be perpendicular to C. It is, however, guaranteed to be perpendicular to the tangent of the curve at $C(t)$. Therefore, the equation to be solved is as follows

$$[C(t) - P)] \cdot C^t(t) = 0 \tag{II.12}$$

where $C^t(t)$ represents the equation of the tangent line to which $\overline{pc}$ is perpendicular. For cubic Bézier curves, Equation II.12 results in a five degree Bézier curve. Equation II.12 is represented as a degree five Bézier in the Bernstein-Bézier form by first letting

$$C_1(t) = C(t) - P \tag{II.13}$$

and rewriting Equation II.12 as

$$C_1(t) \cdot C^t(t) = 0. \tag{II.14}$$

Equation II.13 can be rewritten as

$$C_1(t) = \sum_{i=0}^{n} V_i B_i^n t - P \tag{II.15}$$

where C(t) is written in the Bernstein-Bézier form. Expanding $C^t(t)$ gives

$$C^t(t) = \sum_{i=0}^{n-1} d_1 B_i^{n-1} \tag{II.16}$$

where $d_i = n(V_{i+1} - V_i)$, and the V_i's are the control points of the Bézier curve represented by $C^t(t)$.

Equation II.14 can now be rewritten as

$$\begin{aligned} 0 &= (\sum_{i=0}^{n} V_i B_i^n(t) - P) \cdot \sum_{j=0}^{n} d_j B_j^{n-1}(t) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n} (V_i - P) d_j B_i^n(t) B_j^{n-1}(t) \\ &= \sum_{i=0}^{n}\sum_{j=0}^{n} (V_i - P) d_j z_{i,j} B_{i+j}^{2n-1}(t) \end{aligned} \tag{II.17}$$

where $z_{i,j} = \frac{\binom{n}{i}\binom{n-i}{j}}{\binom{2n-1}{i+j}}$.

Equation II.17 provides the fifth order Bernstein-Bézier representation of the polynomial that must be minimized in order to determine the line between P and C. Given the fifth order Bézier curve, algorithmic root-finding methods exist that can identify the roots for Equation II.17. An example can be found in Glassner (1993).

There are difficulties with using the described approach to solve the nearest-point-on-curve problem for cubic splines. First, the approach requires knowledge of the spline's derivative, which may not be known or may be computationally expensive to obtain. Second, even after the fifth degree Bézier is determined, the speed and efficiency of the method is dependent of the root finding procedure used. Therefore, it appears that a different method should be used to solve the nearest-point-on-curve problem.

Fortunately, other methods exist for solving the nearest-point-on-curve problem that are perhaps more straightforward than implementing a root finding algorithm for a fifth order Bézier. One such method was introduced in Lyche et al. (2002). Lyche et al.'s method is an online approach that does not rely on root finding, and instead uses a two-stage optimization technique that combines the quadratic method and Newton's method to solve the nearest-point-on-curve problem. In Lyche et al., a two-stage optimization technique is applied to a distance equation given in two-dimensions by

$$D(t) = (x(t) - x_0)^2 + (y(t) - y_0)^2 \tag{II.18}$$

where $x(t)$ and $y(t)$ are the parameterized component functions of a cubic spline. Lyche et al.'s method attempts to minimize $D(t)$ through optimization in order to find t^*, the parameter that minimizes $D(t)$. Once the t^* parameter is known, $x(t^*)$ and $y(t^*)$ can be solved to determine the x and y coordinates of the point on the spline that is closest to P. Lyche et al.'s two-stage approach attempts to overcome the shortcomings of Newton's method and quadratic minimization when used on their own. For example, quadratic minimization is known to have a slow rate of convergence, and in some cases, Newton's method is not even guaranteed to converge, especially if a poor initial estimate is used. However, by using quadratic optimization to provide

initial estimates to Newton's method, very fast convergence rates can be achieved (Lyche et al., 2002).

While fast convergence is a possibility, it is not guaranteed. Even though, Lyche et al.. never encountered scenarios where their algorithm did not converge, this does not mean that such cases do not exist. Therefore, while this method shows much promise as a real-time approach, its safety is debatable. However, attempting to minimize the distance function, $D(t)$, is a worthwhile approach. Therefore, a similar approach that attempts to minimize $D(t)$, with guaranteed convergence, will be discussed in Chapter III.2.

II.4.4 Conclusion

Previous research has provided conflicting views concerning the efficacy of touch-based interaction versus mouse-based interaction. On the one hand, research has shown that touch-based interaction is faster than using a mouse, even though it may produce more errors. Other research, particularly that seen in Meyer et al. (1994) has concluded that touch-based interaction is inferior to mouse-based interaction on all levels. Therefore, it is important to determine whether or not touch-based interaction is a detriment to drawing tasks within mobile interaction when compared to mouse-based interaction. If it is determined that drawing using touch-based interaction is indeed inferior, then future design efforts must focus on closing the gap between touch and mouse-based interaction for mobile platforms, since a mouse is not applicable to mobile interaction.

Several methods for improving finger-based drawing were discussed; however, it seems like the only applicable methods are zooming or using the Take-Off technique. The drawbacks to zooming were discussed and it has been shown that Take-Off can result in increased error. Therefore, if touch-based drawing is shown to be quantifiably deficient to mouse-based drawing, techniques for improving touch-based drawing will have to be considered.

The number of samples generated from a user's geospatial input is dependent on the speed with which the user supplies the input. If the user supplies this input at a relatively low speed, points can be included in the input data set that are not necessary for adequately describing the line to be created from the user's input. These unnecessary points consume memory in internal data structures and provide additional computational overhead. They also result in on-screen clutter. Therefore, methods were proposed that may reduce the input data set, while representing the input data set with sufficient accuracy. These methods can be placed into two categories: linear methods and curve methods, both of which were described. Linear methods involve reducing the input data set based on slope or iterative sampling. Curve methods attempt to reduce the set by representing sequences of points using Bézier curves.

Methods for determining the error between the reduced data set and the input data set were also discussed. In the linear method case, determining this error is fairly straight-forward. For curve methods, determining this error requires solving the nearest-point-on-curve problem, and two methods for solving this problem were

introduced. One method involves constructing a fifth order Bézier curve and solving for its roots, which can be time consuming and difficult. A method proposed in Lyche et al. (2002) attempts to minimize a distance function in order to solve the nearest-point-on-curve problem. However, even though the method proposed in Lyche et al. (2002) can converge rapidly, its convergence is not guaranteed. Therefore, alternative methods will have to be considered if convergence is desired.

CHAPTER III

System Description

III.1 The Mobile, Map-Based Tasking Interface

III.1.1 Introduction

The development of the Mobile Map-Based Tasking Interface (MMBTI) has been an iterative process. While development is ongoing, three distinct versions of the MMBTI serve as particularly important design milestones. These versions include: the original interface, which was developed to support mouse-based interaction; a Multi-Touch interface that expanded the original interface by incorporating features specifically designed to support touch-based interaction; and finally, the MMBTI as it exists today. Throughout the discussion, various features supported by the MMBTI will be introduced. The justification for including these features will also be provided.

III.1.2 Previous Interface Design

The MMBTI was developed based on a pre-existing map-based tasking interface that was designed for desktop use. This pre-existing interface made use of traditional input methods, such as a mouse and keyboard, and will be referred to as the Traditional Interface (TI) for the remainder of this work[2].

The TI, shown in Figure III.1, can be divided into three main widgets. The map in the top left and center, displays the robots' location as well as the visual items used for task creation. The right side of the interface contains status information. Three tabs organize this information: By Task, By Robot, and Robot Status. The bottom widget, the Create Task Panel, is for task creation, which displays all the options necessary to create one of the three robot task types: Cross Danger Area, Go to X, and Recon. The Go To X task is classified as a path task, meaning the user specified a path that the robots must follow from start to finish. Cross Danger Area and Recon are area tasks, meaning the user must specify an area of interest to the robots on the map.

Once the TI was designed, it was adapted to support mobile interaction. In order to support mobile interaction, the TI utilizes the built-in touch functionality of a Dell Inspiron XT2 tablet PC (see Figure III.2) running Windows© XP. The Dell XT2 translates touch input into mouse commands. The interface receives these mouse commands and does not deal with any touch input directly. The only gesture that the TI can use consistently is the Point-to-Click gesture. When a user places one finger on the screen, it acts as a mouse-down event. When the finger is lifted, a mouse-up event is triggered. The mouse-up event can be used for selecting options from menus and navigating the on-screen map. No touch or Multi-Touch events were

[2]The TI was developed by Sean Hayes of the Vanderbilt University Human-Machine Teaming Laboratory.

Figure III.1: The Traditional Interface displaying three robots performing a Cross Danger Area (CDA) task.

integrated into the TI. In other words, in order to support touch interaction, the TI converts all touch events into mouse events.

The TI also used traditional widgets to facilitate user interaction. For example, push buttons and combo boxes were present in the TI. While these traditional widgets are suitable for mouse-based interaction, they do not necessarily maintain their usability when subjected to finger-based interaction. For example, push buttons that were suitably sized to be used with a mouse may be too small for finger-based interaction. Widgets may also be placed too close together, causing users to unintentionally interact with the wrong widget.

In order to specify tasks using the TI, the user performs two actions. The first requires the use of the previously mentioned button and combo box widgets to specify task-specific information. Task-specific information is supplied to the interface through the Create Task Panel (see Figure III.3). In addition to supplying task-specific information, the user must also specify the geospatial location where the task will occur. Specifying location is handled through interacting directly with the TI's on-screen map (see Figure III.1).

Task locations can be classified as either paths or areas, both are comprised of waypoints and are specified in the same manner. Areas are closed polygons that are created on the map by using the mouse to specify a waypoint at the location of the mouse click. The resulting polygon is created by joining these waypoints with straight lines. Lines are added sequentially from waypoint to waypoint until a closed polygon is created using

Figure III.2: Touch input on a Dell XT2 running the TI.

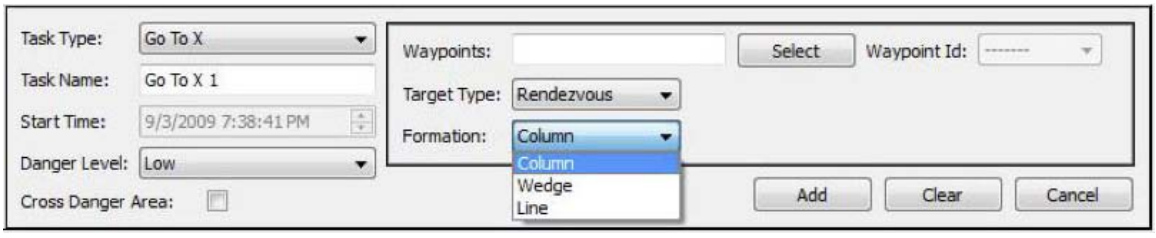


Figure III.3: The Create Task Panel. Note the various pull down menus, text fields, check boxes, and buttons used. These components can make touch-based interaction difficult.

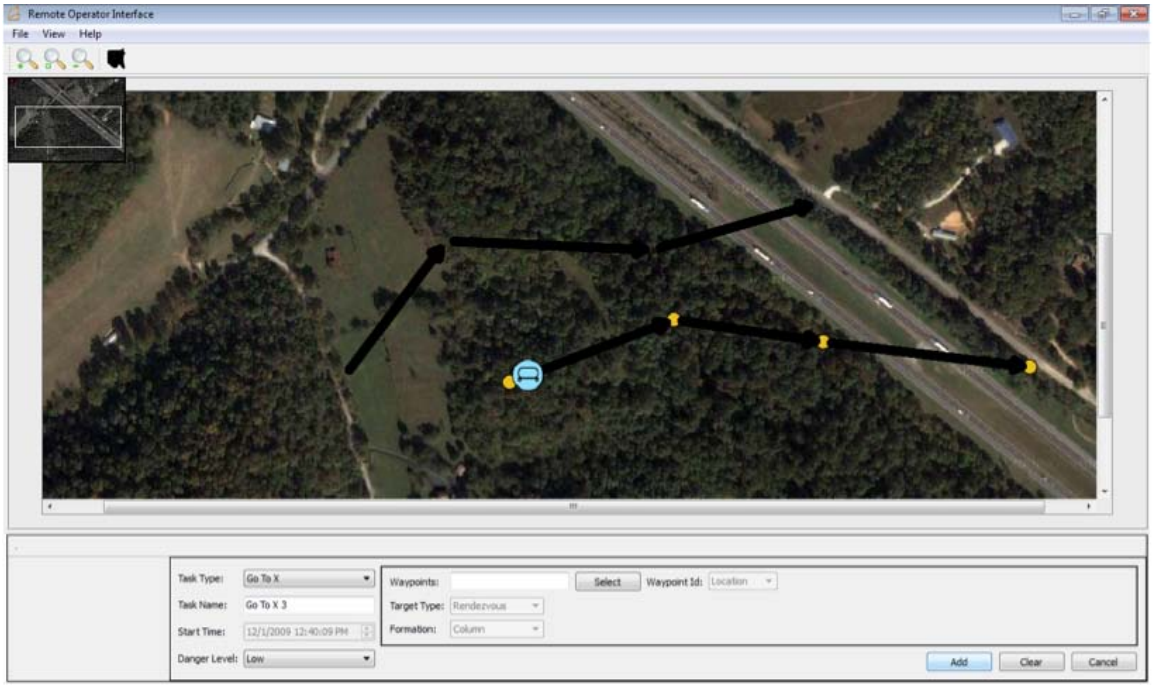


Figure III.4: The TI visualizing a Go To X on screen. Note the black connecting lines joining each waypoint (colored yellow) on the screen. Completed tasks are visualized without waypoints.

the specified waypoints as vertices of the polygon. Paths are specified in a similar manner to areas, except that the last waypoint is not joined by a line to the first, thus a path is created instead of an area. An example of a path specification for the Go To X task can be seen in Figure III.4.

While the TI served as an initial attempt to design a mobile, map-based interface, many design decisions were inappropriate. For example, the lack of Multi-Touch interaction severely hindered the expressive nature of finger-based interaction, which can utilize gestures to perform complex operations in Multi-Touch enabled interfaces. Also, the use of traditional widgets, while appropriate for mouse-based interaction, are typically inappropriate for touch-based interaction. Finally, specifying each individual waypoint for area and path creation is sufficient for paths and areas that require a small number of vertices, but more complex paths and areas (i.e., those that require a large number of waypoints) can be very tedious to create when each waypoint must be individually specified. These deficiencies in touch-based interaction led to the decision that the TI needed to be improved in order to adequately support touch-based (and mobile) interaction.

III.1.3 Multi-Touch Interface Design

The TI's approach to touch-based interaction was rudimentary at best. A new interface was designed to be more accommodating to touch input by incorporating Multi-Touch capabilities. Rather than simply converting all touch input to mouse events, the Multi-Touch Interface (MTI) implements an event system that allows

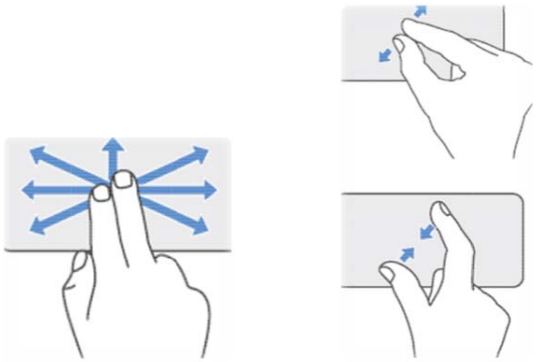


Figure III.5: A visual depiction of the pan (left) and pinch (right-top and right-bottom) Multi-Touch gestures.

for gesture recognition from finger-based input[3]. For example, common gestures include: sliding a finger to scroll, pinching two fingers together to zoom in, pulling two fingers apart to zoom out, and rotating two fingers to facilitate on-screen rotation. More advanced Multi-Touch gestures can also account for the momentum and acceleration used when performing the gesture. These factors can then illicit different responses from the interface (i.e., faster and slower zooming and scrolling). A visual summary of the gesture functionality implemented in the MTI can be seen in Figure III.5.

Aside from Multi-Touch gestures, other functionality was implemented to increase the usability of the MTI. For example, a new method of task location specification was developed. Waypoint specification in the TI, requires the user to touch the screen at the location where the waypoint is to be placed. While this type of interaction is relatively easy for simple tasks (e.g., specifying a straight-line path only requires two waypoints, the start of the line and its end), it can become tedious for more complex paths. For example, if the user wishes to specify a path along a winding road, the user has to add several waypoints at every curve in the road in order to specify the path accurately.

A new means of task input was designed in order to remedy this difficulty. This new method involves treating the on-screen map similar to a canvas and is referred to as the Drawing Mode[4]. The Drawing Mode allows a user to simply "draw" paths on the screen with his or her finger to task the robot. The user initiates a specific task, such as a Go to X task, and then uses a single finger to draw a path from a start point to the robot's desired destination. The interface converts the drawn line in real time and displays waypoints and connecting lines on the map that correspond to the user's drawn line (see Figure III.6).

The Drawing Mode can also be used to specify areas on the screen. For example, in the case of the Recon Area and Cross Danger Area tasks, the robot must complete the task based on an area specified by

[3]Sean Hayes assisted in the development of the MTI.
[4]The Drawing Mode was developed for the MTI by Eli Hooten.

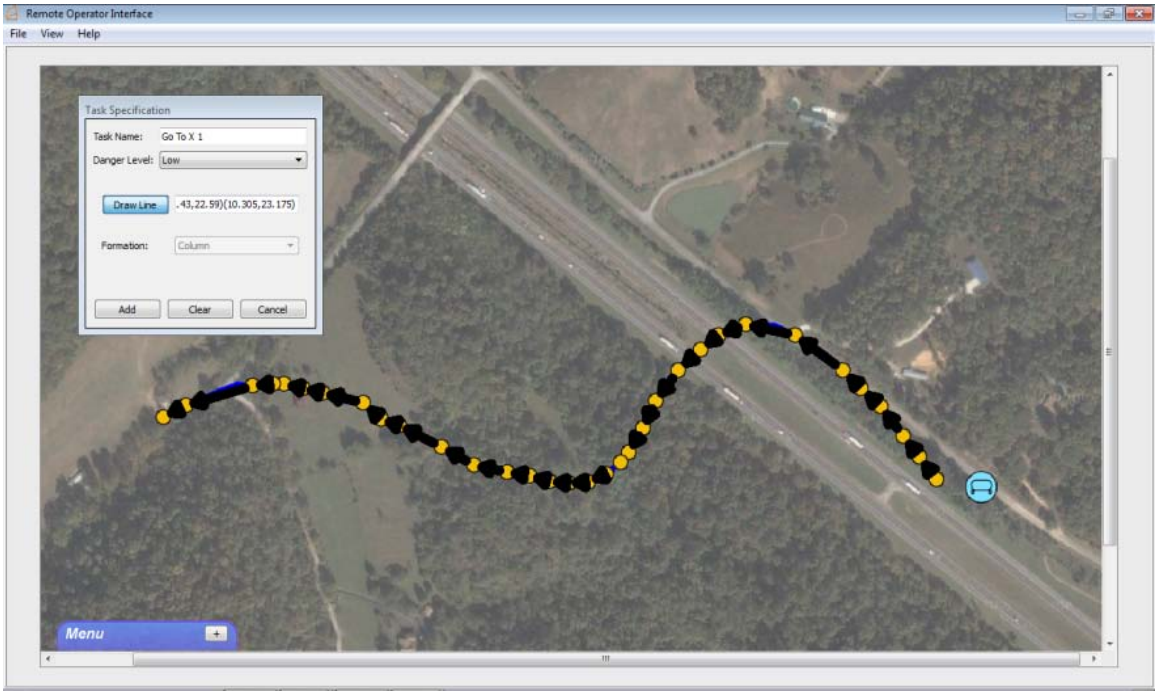


Figure III.6: The MTI displaying a task that was created using the Drawing Mode.

the user. Using the Drawing Mode, a user draws a shape on the screen and the interface converts the shape into a series of waypoints and connecting lines on the screen (see III.7a). This method of shape drawing is particularly beneficial in the event that the user needs to specify a complex area. Once the task is added, the shape fills with a semi-transparent color in order to differentiate line-based tasks from those that require an area specification (see III.7b).

The Drawing Mode is beneficial in that it allows a user to draw complex paths and areas in a very natural manner. Rather than specify single points, whole lines or areas can be specified in one continuous motion. However, the Drawing Mode is not without its drawbacks. For example, waypoints are created as user input is sampled. Sampling user input as it is drawn poses two principal concerns. First, since sampling is limited by the sampling rate of the touch screen and the computational overhead of the interface, sampled points become dependent on the speed with which the user draws. As drawing speed slows, more waypoints are created in that region of the line or area being drawn. The effect of drawing slowly can be seen in the concave section of the area in Figure III.7. Second, the Drawing Mode tends to generate many more waypoints than necessary to represent the drawn line or area. These additional waypoints serve to clutter the map and provide unnecessary extra information that must be stored and managed by the interface. Therefore, the Drawing Mode can benefit from supplying methods that reduce the number of waypoints representing the drawn line or area, while still maintaining an accurate representation of the input.

(a) A shape that has been interpolated into a series of waypoints and connecting lines

(b) Once the task is added, the inside of the shape is filled with a color to help differentiate it from paths.

Figure III.7: A Cross Danger Area task shown in (a) Drawing Mode and (b) after the task is committed.

Figure III.8: The Comms Mode being displayed with several annotations on the screen. The user can choose to display or hide these annotations.

The Communications Mode[5], another mode that utilizes the drawing concept, was also added to the interface. The Communications Mode was introduced as a means for the user to communicate information to other individuals who may be nearby. For example, the user can draw on the screen to indicate areas of interest, potential routes for the robots to travel, or even handwritten notes (see Figure III.8). The user creates these markings directly on the map, which can be hidden, shown, or cleared at any time. An added benefit to the Communications Mode is that it can be active while the user is tasking the robot. Therefore, a user can reference Communications Mode information while specifying tasks.

The MTI provided the initial improvements to the TI to allow for true Multi-Touch based interaction. The MTI also introduced the Drawing Mode for providing geospatial input to the interface. While the Drawing Mode was not without its faults, it still provided a natural method of specifying complex paths and areas. The Communications mode was also introduced in the MTI. The Communications Mode provided a rudimentary means of communicating information through on-screen annotations.

Even though the MTI introduced a number of improvements and new features, it is not considered an adequate interface for mHRI. For example, users still had no efficient means of receiving information. In order to specify a task, the user had to receive the task information to specify through external means, such as locations sketched on a paper map or directions delivered through verbal cues. While these two approaches

[5]The Communications Mode was developed for the MTI by Eli Hooten.

are certainly feasible, they are by no means desirable. An ideal mobile interface delivers all necessary information through the interface, rather than through external means. Therefore, the MTI cannot be considered an interface appropriate for mHRI.

III.1.4 The Communications Mode

The Communications Mode allows users to draw on the screen in order to make notes and indicate areas of interest. The MTI's Communications Mode allowed users to draw task-specific information that can be recalled later to assist in task completion. This mode was shown to users during a prior evaluation (Hayes et al., 2010) and some used it to facilitate task completion; however, created drawings were crude and simplistic, which can limit the usefulness of the mode.

It was determined that, in order to support mHRI, the Communications Mode can be extended to provide task-specific information to users. The benefit to this is obvious; if the interface is used to convey task-specific information, no external means have to be used. Therefore, the user can devote more time and cognitive effort to the environment and the interface. However, due to the limited capabilities of the MTI's Communications Mode, it is insufficient to support the communication of task-specific information to the user. Therefore, the Communications Mode was expanded with greater functionality to support a mobile user performing an mHRI task and implemented within the MMBTI.

Rather than representing task information as crudely drawn shapes on the map, stylized task representations distinguish task information from hand-drawn information that may be supplied by the user. Each task has a specific look and color scheme. In addition, task visualizations are presented with task labels. These labels present task-specific information such as: the task name, the formation a robot team will use when completing the task, and the danger level associated with the task. Users can collapse task labels by using a single-finger touch gesture on the label. Collapsed task labels are minimized to display only the task name. Task labels are also semi-transparent when collapsed, so that the user can see the portion of the map underneath the label. A task label and a collapsed task label can be seen in Figures III.9a and III.9b, respectively.

The Go To X task was represented as a white line with a black border. Chevrons are placed on the line in order to indicate the intended direction of robot movement during the task. The endpoint of the task is also visualized using a white X shape with a black border. An example of the Go-to-X task can be seen in Figure III.10a.

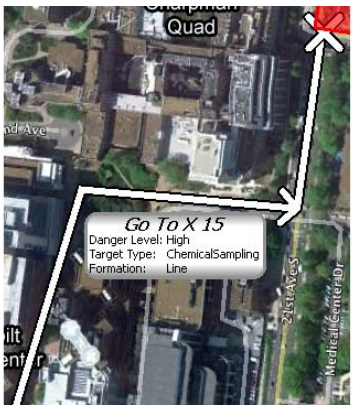
The Cross Danger Area task and Recon task are represented in a similar manner. Each uses a specific color (red for Cross Danger tasks and blue for Recon tasks) to represent task location areas on the map. The color is semi-transparent, such that the map can be seen underneath the task area. Each task also utilizes a

(a) A Go To X task label

(b) A collapsed view of a Recon task label

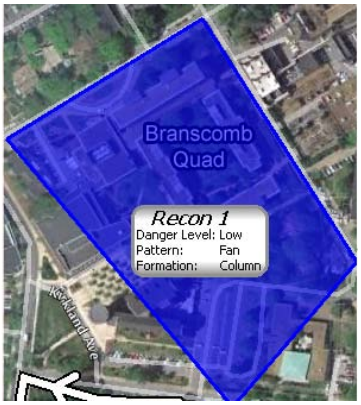
Figure III.9: Task labels as represented in the Communications Mode of the MMBTI.

task label placed at the centroid of the area to display task-specific information. The visual representation of geospatial task information and the corresponding task label together form a task display.



(a) Go To X

(b) Cross Danger Area



(c) Recon

Figure III.10: Stylized task displays as seen in the Communications Mode of the MMBTI.

Much like the MTI's Communications Mode, the MMBTI allows users to create tasks while the Communications Mode is active. Therefore, users can create tasks directly over displayed task-specific information. Supplying task-specific information in this manner removes the need for a user to consult other sources for task-specific information. The Communications Mode can also be enabled or disabled at any time, such that a user can remove and recall task information displays whenever necessary.

A potential drawback of the Communications Mode is the potential for screen clutter while the mode is active. For example, if many tasks are displayed at once, confusion can arise and the user may have difficulty correctly assigning tasks. The difficulty in task assignment is especially prevalent in the case of task displays that overlap. For example, if a Go To X task and a Recon task overlap, the user may accidentally assign the

wrong shared properties (e.g., the danger level, which is a property that all tasks have) of the Recon task to the Go To X task. Therefore, a method needs to be provided to reduce on-screen clutter whenever possible.

The MMBTI's Communications Mode reduced on-screen clutter in two ways. Task labels can be collapsed, which permits the viewing of additional task labels that may have been occluded by task overlap. The Communications Menu also utilizes layering in order to reduce the number of task displays that are on the screen at any given time. Multiple tasks can be present in a single layer that can be shown or hidden at any time.

Layering serves two purposes in the MMBTI. First, it allows multiple task displays to be hidden and displayed simultaneously. Therefore, time does not have to be spent hiding or showing individual tasks. Second, and more importantly, layering provides a mechanism for logically grouping task displays together. For example, suppose a suspicious object was discovered in a parking garage and an operator needs to dispatch a team of robots to investigate the object. The process of moving the robots across the map to the parking garage can require a sequence of tasks, such as a Cross Danger Area to negotiate a hazardous region between the robots and the parking garage, a Go To X task to move from the hazardous region to the parking garage, and finally a Recon task to find the object in the parking garage. The Communications Menu allows these three tasks to be present on a single layer. Since all three tasks are necessary to investigate the suspicious object, the user will need to view and assign all three tasks sequentially; a procedure that is easily accomplished given that all three tasks reside in the same layer. After the tasking is complete, the user can simply hide the layer and remove it from the map, since it may no longer be of importance after tasks have been assigned.

Within the MMBTI, layers are referred to as Task Sets. Task Sets are utilized to represent tasks that logically belong together, and only one can be viewed at a time. The use of Task Sets allows the user to perform sequences of tasks that have related and clear goals, such as performing a series of tasks on the map in order to move to and investigate a suspicious object. Since only one Task Set can be viewed at a time, the user is able to focus on the goal being accomplished by that particular Task Set. Task Sets also help to reduce clutter by reducing the number of tasks that are visible on the screen at any given time. While it is possible to have a Task Set with many task displays, this is typically not the norm and the use of cluttered Task Sets is generally discouraged. An example Task Set can be seen in Figure III.11

In addition to Task Sets, the Communications Mode also contains layers that can hold annotations provided by users, called Drawing Layers. Therefore the original functionality of the MTI's Communications Mode is preserved.

Within the MMBTI, the Communications Mode is managed through use of the Communications Menu. The Communications Menu provides a widget for selecting from the various Task Sets and Drawing Layers that may be present (see Figure III.12) . The Communications Menu groups layers into Task Sets and Drawing

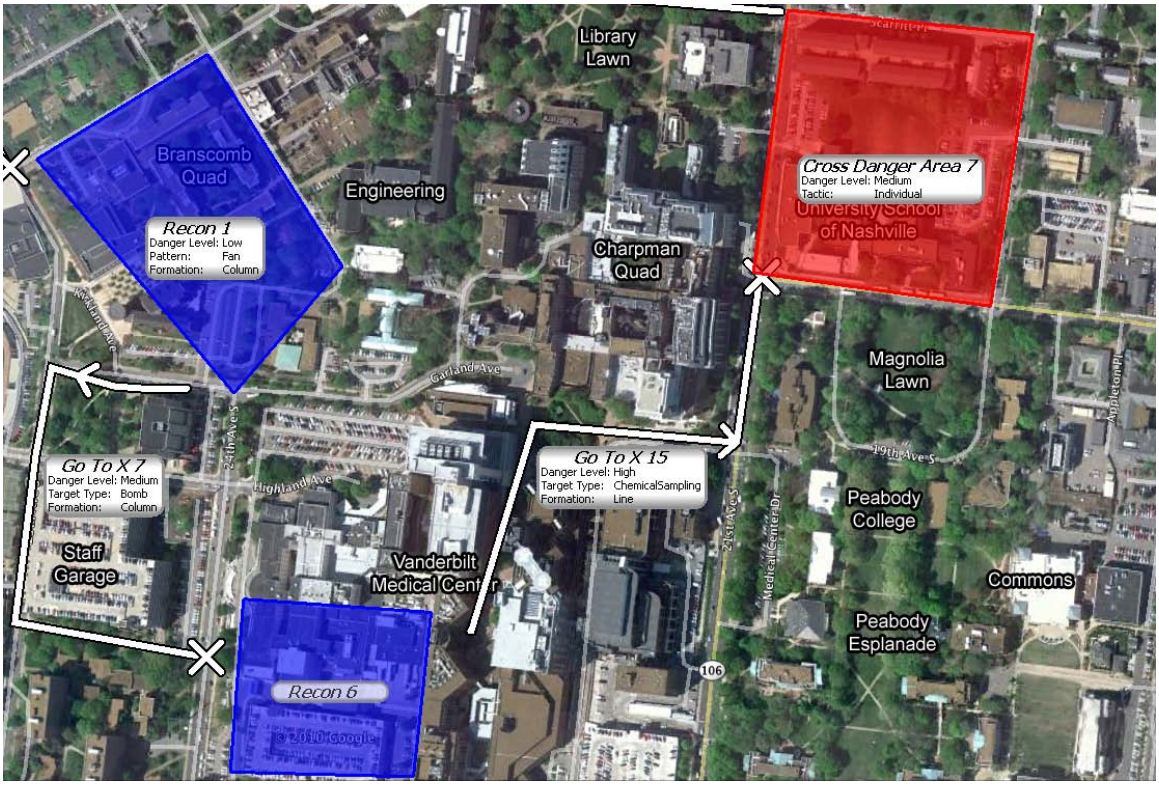


Figure III.11: An example Task Set as displayed in the MMBTI.

Layers. Since only one Task Set or Drawing Layer may be visible at a time, selecting one of the layers via a push button causes the currently active layer to be hidden and the selected layer to be shown. If no layer is active, the selected layer is simply displayed.

III.1.5 Mobile, Map-Based Tasking Interface Design Summary

The design of the MMBTI was an iterative process where incremental improvements were added at each phase of the design. The first attempt at a mobile interface was the TI, which introduced very rudimentary handling of touch input by converting all user touch-based input into mouse events. The TI did nothing to leverage the expressivity of human fingers as an input method.

The deficiencies in the TI led to the development of the MTI. The MTI implemented Multi-Touch functionality which allowed the user to interact with the interface using Multi-Touch gestures. The MTI also introduced a Drawing Mode, which allowed for the specification of complex paths and areas in a natural manner. In addition to the Drawing Mode, the MTI also introduced the Communications Mode. The Communications Mode within the MTI provided a means for a user to make annotations on the map and recall them at a later time. Despite the improvements of the MTI, it was still unsuitable for mHRI tasks. For example, the MTI required the use of external sources (such as paper-based maps and audio cues) to provide

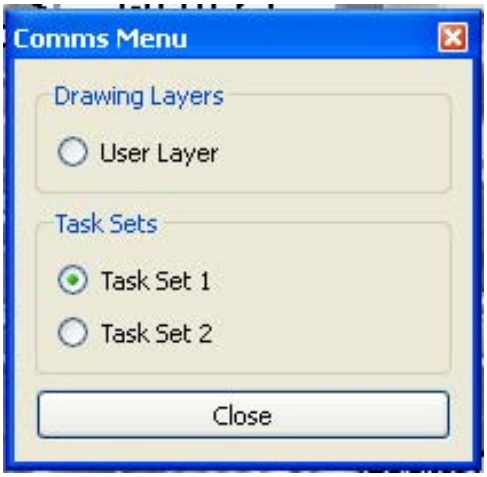


Figure III.12: The Communications Menu.

task-specific information, thus hindering it from being a standalone tool for mHRI tasks.

The MMBTI expanded the functionality of the Communications Mode in order to become viable for mHRI tasks. The MMBTI's Communications Mode provided stylized representations of tasks within logical groupings known as Task Sets. These Task Sets helped to reduce on-screen clutter and provide a focused means of accomplishing goals within the MMBTI. The MMBTI's Communications Mode also retained the annotation functionality from the MTI by utilizing Drawing Layers.

It is believed that the MMBTI's Communications Mode will allow it to be fully suitable for mHRI tasks. However, before this can be ascertained, a usability study of the Communications Mode must first be conducted. A user evaluation comparing the Communications Mode to traditional modes of delivering task-specific information (e.g., providing task-specific information on a paper map or delivering information through verbal cues) was conducted to determine whether or not the Communications Mode is superior to traditional input methods.

III.2 Reducing Geospatial User Input

To investigate the reduction of user geospatial input, four methods (two linear methods and two curve methods) were developed. This section will describe each of the four developed reduction methods. The four methods developed include: Slope reduction, Iterative reduction, Cubic reduction, and Iterative Cubic reduction. For each of the methods, equations developed in Chapter II.4 will be leveraged.

Each of the developed reduction methods work in an online fashion. In other words, they reduce the input data set as the user is drawing. Therefore, the methods can be applied in real-time rather than waiting for the user to finish drawing a line and then reducing it, altering the line's final appearance on screen.

III.2.1 Linear Methods for Reducing User Input

The linear methods evaluated are very similar to those described in Chapter II.4.3.1. Two methods were designed: Slope reduction and Iterative reduction. Slope reduction works by using Equation II.1 to determine the slope between the last input point, $\mathbf{P}_{n-1}$, and the current point, $\mathbf{P}_n$. Once this slope is determined, it is compared to the slope that was calculated at the last iteration (the slope between points $\mathbf{P}_{n-2}$ and $\mathbf{P}_{n-1}$). If the difference between the two slopes does not exceed a predefined threshold, point $\mathbf{P}_n$ is removed from the input set.

For the test application, various slope thresholds were tested by drawing many lines using the Slope reduction method. Through experimentation, it was determined that a suitable slope threshold was 0.05. However, this threshold may vary depending on the desired application of the method. A threshold of zero could theoretically be used. Using a threshold of zero ensures that point $\mathbf{P}_n$ is collinear to $\mathbf{P}_{n-1}$ and $\mathbf{P}_{n-2}$; however, in practice it is extremely difficult to draw a long line segment with a constant slope, meaning very little reduction of the input data set will occur. Therefore, a slightly higher threshold still provides some reduction, while preserving the integrity of the input data set. Slope reduction is defined by Algorithm 1.

Algorithm 1: The Slope reduction algorithm

input : A slope threshold, threshold; the previously calculated slope previousSlope; the input set $\mathbf{P}$; the last point in $\mathbf{P}$, $\mathbf{P}_n$; the sampled point p
output: $\mathbf{P}$

slope $\leftarrow$`CalculateSlope(`$\mathbf{P}_n$, p`)`
if slope $-$ previousSlope $>$ threshold **then**
  $\mathbf{P} \leftarrow \mathbf{P} + \langle p \rangle$

The other linear method developed was Iterative reduction. Iterative reduction uses a predefined removal index, k. A counter is initialized to zero and incremented every time an input point is sampled. When the count reaches k the input point sampled at that time step is added to the input data set and the counter is reset to zero. Iterative reduction is defined in Algorithm 2.

The removal index k is a predefined heuristic in Algorithm 2. For the purpose of this thesis, experimenta-

Algorithm 2: The Iterative reduction algorithm

input : the input set $\mathbf{P}$; the sampled point p; the removal index k
output: $\mathbf{P}$

static counter $\leftarrow 0$
counter $\leftarrow$ counter + 1
if counter $=$ k **then**
  $\mathbf{P} \leftarrow \mathbf{P} + \langle p \rangle$
  counter $\leftarrow 0$

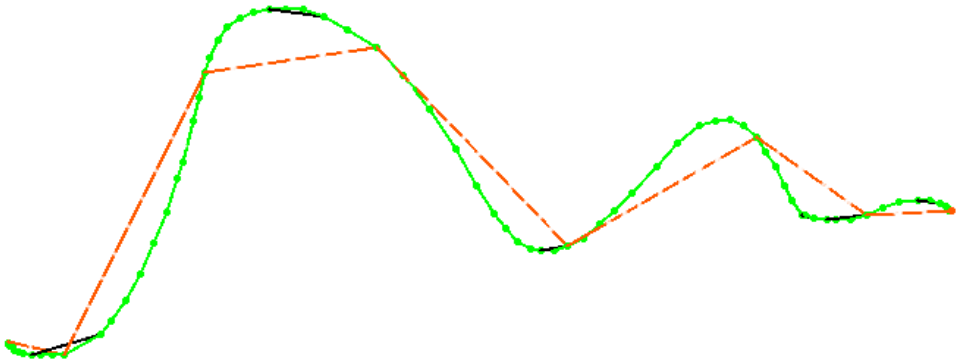


Figure III.13: Slope (black) and Iterative (orange dashed) reduction applied to an input data set (green). The Slope reduced line is partially obscured by the input data set.

tion indicated that a suitable k value would be 10; however, it is expected that the value of k can vary wildly depending on the application, and even vary amongst different users for the same application. Iterative reduction provides a guaranteed reduction on the input set. Given any input set, it is guaranteed to be reduced by a factor of k. Even though the rate of reduction is constant, it can have wildly differing effects depending on the drawing speed of the user generating the input set. For example, a user who draws very slowly may benefit significantly from the Iterative reduction method as many points in the set are likely to be unnecessary. However, if a user draws quickly, valuable information can be lost via Iterative reduction. Due to the dependency on drawing speed, it is speculated that the success of Iterative reduction will vary greatly from user to user.

Iterative and Slope reduciton represent two linear methods for data set reduction. Figure III.13 visually demonstrates the linear reduction methods. In Figure III.13 points in the user's input data set are shown in green and each point is connected to the next point in the set by a solid green line, forming a continuous line composed of linear segments. Slope reduction is represented by the black line, and is partially obscured by the user's input line. Iterative reduction is displayed in orange. The user input was supplied at a very high rate of speed to supply a limited data set with a large spacing between data points in order to exaggerate the effects of the reduction methods.

III.2.2 Curve Methods for Reducing User Input

Two curve methods for reducing user input were developed: Cubic reduction and Iterative Cubic reduction. Cubic reduction samples every set of four points in the input data set and uses the first and last point as the start and endpoints of a cubic Bézier. The second and third points are used as control points. Sampling is repeated for every four points, using the endpoint from the last iteration as the start point for the current iteration, which guarantees that the cubic Bézier curves generated form a cubic spline.

Utilizing the second and third points as control points has advantages and disadvantages. One advantage is that, since control points are sampled directly from the input set, there is no need to perform the calculations necessary to construct the cubic Bézier, which reduces overhead. The reduction of overhead is crucial to ensure the algorithm operates in real time. Another advantage is that for every four points, two are removed from the map, potentially reducing screen clutter.

The biggest disadvantage is that the reduction of the data set is somewhat superficial. Since all points in the input set are used to construct the cubic spline, none of the points can be discarded. It is possible that after the spline is created and rendered on screen, the control points can be discarded, thus reducing the input data set. However, in the event that the data set is needed again, calculations will have to be performed to obtain the discarded control points. Another disadvantage is that, since points in the actual data set are used as control points, a small amount of error is guaranteed. Error is guaranteed because a cubic Bézier does not pass through its control points, therefore control points used to make the Bézier are guaranteed to be off the curve and, thus, have an error value associated with them. However, depending on the cubic Bézier, control points can be very close to the curve and have little error.

The construction of each cubic Bézier is application specific, therefore an algorithm is not provided. In the case of this thesis, each cubic segment was calculated using a direct application of Equation II.3, which proved to be efficient enough.

Iterative Cubic reduction is essentially a combination of the Cubic method and the Iterative method. Rather than sampling sets of four points and using all four to calculate a Bézier curve, the Iterative Cubic method samples a specified set of k points and attempts to interpolate a cubic Bézier that passes through all k points. Note that depending on the size of k and the placement of the k points, it is not guaranteed that one cubic Bézier curve can pass through all points since we are trying to fit all points with a single cubic Bézier.

The Iterative Cubic approach to curve fitting is rather rudimentary, but requires little computation overhead. Of the k points sampled, the Iterative Cubic method selects the $\frac{k}{4}$th point and the $\frac{3k}{4}$th to use as control points. In the event that $\frac{k}{4}$ and $\frac{3k}{4}$ do not evaluate to whole integers, the nearest whole integer is used. The logic behind choosing the $\frac{k}{4}$th point and the $\frac{3k}{4}$th points is that since the user is providing drawn input, for small values of k, data points can be spaced relatively close together. Therefore, selecting the $\frac{k}{4}$th point and $\frac{3k}{4}$th points in the data set as control points may form a relatively adequate description of the input. Also, since a smooth curve is generated, the output after reduction may look more natural (i.e., it will be more likely to resemble a continuous, drawn line) than the Iterative method that connects reduced data points using straight lines. Iterative Cubic reduction is defined by Algorithm 3.

Iterative Cubic reduction, like Iterative reduction, provides guaranteed reduction by a factor of $k-4$, since for every k data points, only four are used. Also like Iterative reduction, Iterative Cubic reduction's

Algorithm 3: The Iterative Cubic algorithm.

input : A set of points, **P**, such that $\mathbf{P} = \{p_{n-k}, p_{n-k+1}, \ldots, p_{n-k+(k-1)}, p_n\}$
output: A cubic Bézier curve, C

C $\leftarrow$`CalculateBézierCurve(`$p_{n-k}, p_{n-\frac{k}{4}}, p_{n-\frac{3k}{4}}, p_n$`)`

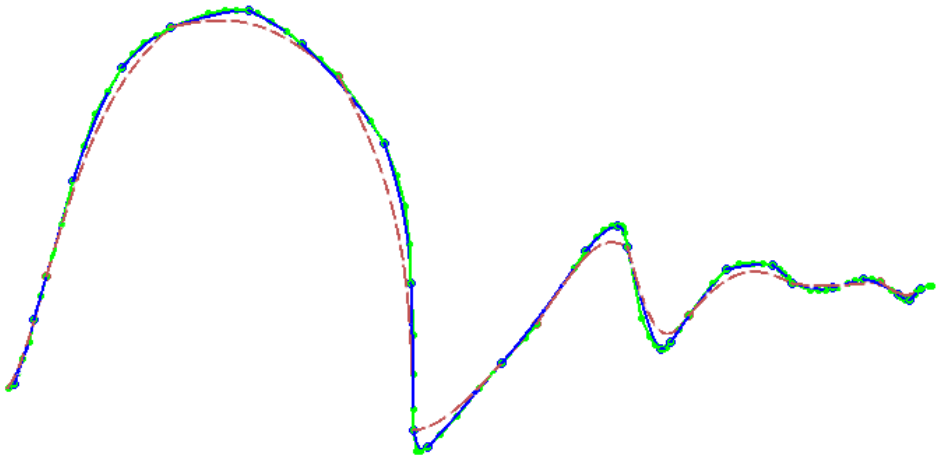


Figure III.14: Cubic reduction (blue) and Iterative Cubic reduction (red dashed) reduce the input state set (green).

effectiveness is dependent on the speed at which the user draws and the value of k. Based on experimentation, a k value of seven generates the most reduced output that still accurately represents the input data set. For this implementation, the cubic Bézier curve was calculated via a direct application of Equation II.3.

Cubic and Iterative Cubic are two curve methods for reducing an input data set. An example of these methods reducing an input set can be seen in Figure III.14.

III.2.3 Error Calculation of Reduced Data Sets

As described in Chapter II.4.3.3, reducing a user's input data set can result in error. This error is quantifiable and can be determined by using the methods mentioned in Chapter II.4.3.3, however, these methods can take a considerable amount of time to converge, if they converge at all. Convergence is crucial for this application, since a reliable error metric is required in order to determine the efficacy of the reduction methods used. Therefore, a new error sampling method, with guaranteed convergence was developed.

The method itself is quite simplistic and leverages the fact that drawn curves can be parameterized. Rather than attempt to minimize a distance function like the method proposed by Lyche et al. (2002), the implemented error method breaks the curve, C, into n discrete parameters and evaluates the distance between $C(n)$ and a point P, as follows

$$D(n) = (C_x(n) - P_x)^2 + (C_y(n) - P_y)^2. \quad \text{(III.1)}$$

The approximate closest point on the curve C is found by repeating Equation III.1 for every n, and the $C(n)$ that corresponds to the smallest distance is taken as the closest point on the curve C to the point P. This method of error calculation also works for linear methods, since C can be any parameterized line. The error calculation algorithm is defined in Algorithm 4.

Algorithm 4: Error calculation algorithm for the nearest-point-on-curve problem.

input : The number of discrete parameters n, point P, and curve C
output: The shortest distance from P to a point on curve C, $D_{\overline{pc}}$

$D_{\overline{pc}} \leftarrow \infty$
for $i \leftarrow 1 : \mathsf{n}$ **do**
  $D \leftarrow (\mathsf{C}_x(\mathsf{n}) - \mathsf{P}_x)^2 + (\mathsf{C}_y(\mathsf{n}) - \mathsf{P}_y)^2$
  if $D < D_{\overline{pc}}$ **then**
    $D_{\overline{pc}} \leftarrow D$

Much like the error calculation methods mentioned in Chapter II.4.3.3, Algorithm 4 has its own drawbacks and advantages. The primary benefit of Algorithm 4 is that it is guaranteed to converge. Therefore, error values will always be known and can be calculated in $O(n)$ time. However, Algorithm 4 does not guarantee that the closest point found on curve C is the actual closest point. Since the algorithm uses n discrete parameter steps, the granularity of the algorithm can be fine tuned. In other words, the larger the n used, the closer $D_{\overline{pc}}$ is to the true distance from P to the closest point on C. However, increasing n increases the running time of the algorithm. Therefore, the implementer must decide on the desired granularity when implementing Algorithm 4. If a rough estimate is all that is required, then a relatively small n (compared to the length of the curve) can be used. If an exact quantity is desired, then a large n must be used. In this case, it is probably preferable to use one of the methods defined in Chapter II.4.3.3, and risk encountering a situation where those methods will converge slowly or not at all.

Algorithm 4 can be optimized. While it will still perform in $O(n)$ time, improvements such as caching all $C(n)$ values ahead of time can speed up the algorithm. If n is known ahead of time, then $C(n)$ can be stored in a lookup table before Algorithm 4 is initiated. The algorithm can also be performed incrementally on line segments as the user draws, so that the error does not have to be calculated on the entire drawn line after the user has finished drawing it. The implementation used for this thesis applies Algorithm 4 to individual line segments as the user draws, and the computational overhead is not noticeable.

Algorithm 4 can also be extended to find the total error between two lines. Finding the error between two lines is useful if, for example, the user is attempting to trace a line or shape displayed on the screen. When

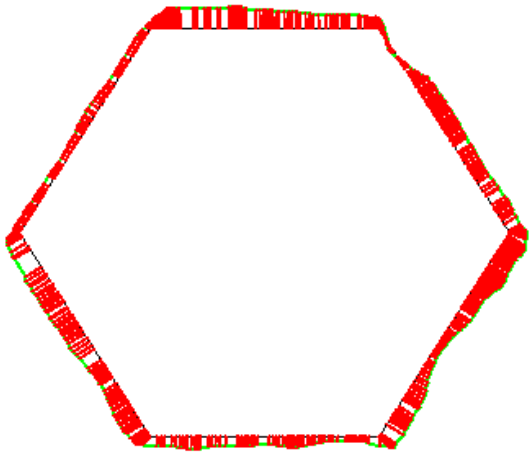


Figure III.15: Shape error (red) between a user's input and a predefined hexagonal shape.

the user is tracing a line, error can be calculated by finding the nearest point on the shape for every point in the user's input data set. However, if there are n discrete segments of the shape, and m points in the input set, Algorithm 4 runs in $O(mn)$ time, which can be quite slow. This slowdown was considered to be acceptable since the error between user input and a shape can be performed offline and not affect the performance of the user. A visual example of calculating the error between a shape and user's drawn input can be seen in Figure III.15.

III.2.4 Reducing User Input Summary

Four online methods have been described for reducing user input: Slope, Iterative, Cubic, and Iterative Cubic. The slope method uses a predefined slope threshold to determine whether or not a point should be included in the reduced data. Iterative reduction reduced the input data set based on a reduction parameter k, and reduction is guaranteed to be a factor of k. However, it is expected that the effectiveness of Iterative reduction will vary from user to user. Cubic reduction does not perform reduction in a true sense, since it uses all of the input data points to form a cubic spline that passes through or near all points in the input data set. Cubic reduction also reduces screen clutter by removing two waypoints from the screen to use as control points. Iterative Cubic reduction attempts to construct a cubic Bézier curve that spans a range of k points. Iterative Cubic reduction suffers from the same drawbacks as Iterative reduction, but provides greater reduction versus Cubic reduction.

An error calculation method was developed to quantify the error incurred by reducing the input data set. The error calculation method is presented in Algorithm 4. The error calculation algorithm has benefits and disadvantages versus the methods described in Chapter II.4.3.3; however, Algorithm 4 was used because it

guarantees convergence in a known running time. Algorithm 4 was also extended to allow it to calculate the error between each point in an input set and an on-screen shape. This extension is beneficial if the user is attempting to trace a shape on the screen, but it takes $O(mn)$ time perform. An $O(nm)$ running time is acceptable for this thesis, since the calculation of error between an input set and a shape does not have to occur in real time.

III.3 System Description Summary

The preceeding section discussed the development of the MMBTI and the Communications Mode. The MMBTI was designed as an iterative process, adding features to support mHRI at every iteration. The Communications Mode presents task-specific information to users to assist in the task creation process. It is believed that the development of the Communications Mode allows the MMBTI to fully support mHRI as a mobile interface.

Methods for reducing user-supplied geospatial input were also described. These methods include: Slope, Iterative, Cubic, and Iterative Cubic. Each method was described in Chapters III.2.1 and III.2.2. A method for calculating the error between a point in the input data set and the reduced data set was developed and can be seen in Algorithm 4. The error sampling method is guaranteed to converge and runs in $O(n)$ time.

CHAPTER IV

Communications Mode Evaluation

IV.1 Introduction

A participant evaluation was performed to investigate the usefulness of the Communications Mode in the MMBTI. The Communications Mode was compared to two other communicative modalities that can be used to present task-specific information to a participant. These methods were: Paper, which presents task-specific information on a paper based map of the environment, and Auditory, which presents task-specific information to the participant via verbal cues. It was hypothesized that the Communications Mode would be comparable, if not superior to the Paper and Auditory modalities.

IV.2 Method

The Communications Mode Evaluation was performed using a Dell XT2 tablet laptop running Windows© 7. The interface was displayed over the full screen at a resolution of 1280×800 px. The MMBTI was developed in C++ and utilized Qt 4.6.3 to create widgets and manage touch gestures. Each participant was asked to hold the tablet with a non-dominant hand and interact with the interface using the dominant hand. Participants were seated and were allowed to rest their arms on their lap or on a table.

Eight participants performed the evaluation. Two were female and six were male. Two were left-handed, five were right-handed, and one person was ambidextrous. Six participants reported using a touch-enabled device at least one hour per week. Participants' average age was 22.13 ± 4.25 years.

Data was obtained through the use of automatic data-logging. Examples of logged data include: task completion times, committed task field values, and the paths and areas the participant specified for each task. Task completion times were used for the purpose of this evaluation, while committed task field values, and path and area information were simply stored.

A ten-minute training session was provided prior to evaluating a new modality. During the training, the participant was familiarized with the interface and specific modality features. The participant performed a series of practice tasks. After practicing, the participant was allowed to explore the interface and modality features until he or she felt comfortable with the interaction. During the training, the participant was allowed to ask questions pertaining to the interface and its operation. After the training, a 20-minute scenario was completed. This process was repeated for each of the three modalities.

Each scenario required participants to input tasks for a team of robots using two task sets. The participants were required to specify up to eight consecutive tasks per task set. An equal number of each task type was

performed across scenarios. The tasks were designed to be similar in difficulty (e.g., similar size areas, similar complexity of shapes, and equal numbers of fields to change) across each modality tested. Different maps were used for each scenario, but each map possessed similar complexity in terms of structure density, street layout, and environmental features.

A with-in subjects evaluation was designed to evaluate the three communicative modalities (i.e., Communications Mode, Paper, and Auditory). The modality presentations were randomized across participants. The independent variable was the communicative modality. Dependent variables were automatically logged objective scenario completion times, subjective modality ratings were assessed with Likert scale questions, NASA TLX subjective workload, and subjective modality rankings.

Scenario completion time was used as an objective metric to evaluate the three modalities. Scenario completion time was measured as the time span between the specification of the first task until the committing of the last task. Using this time span, time intervals, such as interface load and exit time, the time interval required for map familiarization, and the time interval required to open the Task Specification window are excluded. Therefore, the measured duration provides an accurate depiction of the total time the participant spent creating the tasks.

Subjective Likert scale questionnaires were administered for each evaluation condition after the condition trial. The NASA TLX was also administered for each condition trial. A modality ranking questionnaire was administered upon evaluation completion.

The Likert scale questions assessed the following characteristics of each modality: Comprehension, the participant's comprehension of the task as it was presented; Assignment, the participant's ability to correctly task the robots; Placement, the participant's ability to correctly place tasks on the map; and Confidence, the level of confidence experienced by the participant while tasking the robots on a scale of 1 to 9, where 9 is the best.

The Communications Mode presented tasks using Task Sets. The Paper modality provided task sets printed in color on 8.5" x 11" paper, which looked identical to the Communications Mode Task Sets (see Fig. III.11). The Auditory modality provided all task specific information via verbal cues, in which the experimenter told participants the task type, the location of the task, and any task specific settings.

IV.2.1 Results

IV.2.1.1 Objective Measures

The mean completion time was shortest for the Communications Mode, while the Auditory modality resulted in the longest scenario completion time. The scenario completion time means and standard deviations are provided in Table IV.1. A Bonferroni test comparing the Communications Mode and Auditory modality

and the Communications Mode and Paper modalities was performed with a familywise $\alpha_{fw} = 0.05$ and a per comparison $\alpha_{pc} = \frac{0.05}{2} = 0.025$. Using these alphas, a t-test compared the scenario completion times of the Communications Mode and Auditory modality and obtained a significant result ($t_7 = 3.99$, $p < 0.003$). A t-test was also used to compare the scenario completion times of the Communications Mode and Paper modality and the Paper and Auditory modalities. Neither of these comparisons were significant. This lack of significance between the Communications Mode and Paper scenario completion times may be due to both modalities visually presenting the task.

IV.2.1.2 Subjective Measures

The means and standard deviations of the Likert scale evaluation results for Comprehension, Assignment, Placement, and Confidence are presented in Table IV.2. Participants reported the highest mean rating for the Communications Mode across all four categories. The Auditory modality was consistently ranked the lowest. A Bonferroni test comparing Communications Mode and Auditory modality, Communications Mode and Paper modality, and the Paper and Auditory modalities was performed with a familywise $\alpha = 0.05$ and a per comparison $\alpha = \frac{0.05}{3} = 0.167$. Using these alphas, comparisons were made considering Comprehension, Assignment, Placement, and Confidence.

A t-test revealed that Comprehension was rated significantly easier using the Communications Mode when compared to the Auditory modality ($t_7 = 2.97$, $p < 0.011$). Comprehension was also rated significantly easier with the Paper modality when compared to Auditory ($t_7 = 2.81$,$p < 0.013$). Both visual-based modalities (i.e., the Communications Mode and Paper) were rated significantly higher than the Auditory modality, thus indicating that task comprehension was better when a visual map-based modality was used. While the mean Comprehension rating for the Communications Mode was higher than the Paper modality, the difference was not significant. It is suspected that this result is due to the similarities between the Communications Mode and Paper modality; however, these results do show that tasks are easier to comprehend when placed directly on the map.

A t-test revealed that participants found the Communications Mode significantly easier for Assignment when compared to the Auditory modality ($t_7 = 4.79$, $p < 0.001$) and the Paper modality ($t_7 = 3.74$, $p < 0.0037$). Participants also found the Paper modality significantly easier for Assignment than the Auditory

Table IV.1: Scenario completion durations using each modality.

Duration (sec)	Communications Mode	Paper	Auditory
Mean	491.78	640.55	792.97
Std	149.78	353.51	246.84

Table IV.2: Communicative Modality Likert Scale Ratings.

Modality		Comprehension	Assignment	Placement	Confidence
Communications Mode	Mean	8.21	8.66	7.31	7.76
	Std	0.751	0.466	1.01	1.17
Paper	Mean	7.76	7.76	6.3	7.54
	Std	0.825	0.699	1.67	0.955
Auditory	Mean	6.08	5.96	5.4	5.74
	Std	1.65	1.44	1.52	1.17

Table IV.3: NASA TLX subjective workload ratings.

Modality		Score	Mental	Physical	Temporal	Performance	Effort	Frustration
Communications Mode	Mean	51.055	35.250	31.375	35.125	78.000	46.750	22.250
	Std	7.064	15.295	16.017	17.707	15.901	9.114	15.040
Paper	Mean	56.739	47.625	29.750	46.625	76.250	54.625	33.625
	Std	9.236	16.414	17.136	19.500	11.573	12.794	19.420
Auditory	Mean	64.57	58.13	37.13	63.63	63.63	67.25	48.25
	Std	10.02	17.48	19.55	21.90	18.28	14.67	22.61

modality ($t_7 = 4.00$, $p < 0.0026$).

Even though the Communications Mode Placement mean was the highest, a significant difference was not found between the Communications Mode and Paper modality. However, the Placement mean for the Communications Mode was significantly higher when compared to the Auditory modality ($t_7 = 6.96$, $p < 0.011$).

The Confidence means for both the Communications Mode and Paper modality were very similar, with the Communications Mode being rated slightly higher. The Paper modality and the Communications Mode provided a visual reference with which to compare the task being specified. Therefore, participants may have a comparable amount of confidence in either modality due to the ability to visually verify the task specification requirements. However, the Communications Mode Confidence ratings were significantly higher than the Auditory modality ($t_7 = 8.83$, $p < 0.001$).

NASA TLX was used to assess overall workload for each modality. The Communications Mode resulted in the lowest overall mean workload, while the Auditory modality resulted in the highest mean overall workload. Table IV.3 also presents the means and standard deviations for each of the NASA TLX components. Generally speaking, the Auditory modality received the worst mean ratings across the components and the Communications Mode performed better than the Paper modality, except for the Physical component.

A Stepdown (Holms) Bonferroni was used to compare the NASA TLX data in order to maintain an acceptably tight familywise Type 1 error rate with significant power. For the Holms Bonferroni, $\alpha_{fw} = 0.5$. The per comparison alpha, α_{pc}, was set to $\alpha_{pc} = \frac{0.05}{3} = 0.0167$ for the most significant p-value; $\alpha_{pc} = \frac{0.05}{2} = 0.025$ for the next most significant p-value; and $\alpha_{pc} = 0.05$ for the least significant p-value.

A t-test found that the Communication Mode's overall workload was significantly lower than the Auditory modality ($\alpha_{pc} = 0.0167$, $t_7 = 4.12$, $p < 0.0025$). The same significant relationships were found when comparing mental demand ($\alpha_{pc} = 0.025$, $t_7 = 3.27$, $p < 0.007$), temporal demand ($\alpha_{pc} = 0.025$, $t_7 = 3.41$, $p < 0.006$), performance ($\alpha_{pc} = 0.0167$, $t_7 = 3.09$,$0 < 0.009$), effort ($\alpha_{pc} = 0.0167$, $t_7 = 4.78$,$p < 0.0011$), and frustration ($\alpha_{pc} = 0.0167$, $t_7 = 3.70$, $p < 0.004$). Significance for each of the categories indicates that the Communications Mode results in lower workload than the Auditory modality.

A t-test found that for the Paper modality the overall workload was significantly lower than the Auditory modality ($\alpha_{pc} = 0.025$, $t_7 = 3.84$, $p < 0.004$). Significant relationships were also found when comparing mental effort ($\alpha_{pc} = 0.0167$, $t_7 = 3.78$, $p < 0.004$), temporal demand ($\alpha_{pc} = 0.0167$, $t_7 = 3.58$, $p < 0.005$), performance ($\alpha_{pc} = 0.05$, $t_7 = 2.03$, $p < 0.041$), effort ($\alpha_{pc} = 0.025$, $t_7 = 4.38$, $p < 0.002$), and frustration ($\alpha_{pc} = 0.05$, $t_7 = 2.32$, $p < 0.027$). These results for each category indicates that the Paper modality results in lower workload than the Auditory modality. These results combined with the Communications Mode results indicate that visual modalities result in lower workload than auditory-based modalities for map-based tasking.

A t-test was also applied to the NASA TLX ratings between the Communications Mode and Paper modalities. It was determined that the Communications Mode had significantly lower ratings for overall workload ($\alpha_{pc} = 0.05$, $t_7 = 2.28$, $p < 0.03$), mental demand ($\alpha_{pc} = 0.05$, $t_7 = 2.22$, $p < 0.031$), temporal demand ($\alpha_{pc} = 0.05$, $t_7 = 2.17$, $p < 0.034$), effort ($\alpha_{pc} = 0.05$, $t_7 = 2.69$, $p < 0.016$), and frustration ($\alpha_{pc} = 0.25$, $t_7 = 2.49$, $p < 0.021$). These results indicate that the Communications Mode results in lower workload than the Paper modality.

The modality ranking survey revealed that 87.50% (7 of 8) of participants found the Communications Mode the easiest to understand and interpret. All participants indicated that the Communications Mode conveyed the most information about the tasks. All participants indicated that when using the Communications Mode, it was very easy to remember the assigned tasks. Participants also unanimously indicated that the Communications Mode gave them the greatest level of confidence when assigning tasks. The Communications Mode was also unanimously preferred versus the other two modalities.

The Auditory modality was consistently ranked lowest by all participants in every category, except by one participant who thought the Auditory modality was the easiest to understand and interpret. However, this same participant also ranked the Communications Mode modality as the best overall.

IV.2.2 Discussion

It was hypothesized that the Communications Mode would be superior to the Auditory modality and comparable, if not superior to, the Paper modality. The results clearly confirm the hypothesis that the Communications Mode is superior to the Auditory modality for map-based tasking interfaces. Improved comprehension, faster

task completion times, and the unanimous ranking of the Communications Mode as the best, supports the hypothesis that the Communications Mode is at least as good as, if not superior to, the Paper modality. It is also important to note that superiority over the Paper modality is not required for the Communications Mode to be viable for mHRI tasks. Even if the Communications Mode does not offer additional information compared to a Paper modality, the added benefit to the Communications Mode is that it is integrated into the interface. Therefore, the use of an external, paper-based source, such as an annotated map, for task-specific information is not required when the Communications Mode is used. Thus, the Communications Mode is a viable method for supporting mHRI and no benefit will be lost when compared to using a paper-based modality.

The workload data also support the hypothesis that the Communications Mode is superior to the Auditory modality. Comparing Communications Mode and Auditory NASA TLX ratings revealed that the Communications Mode was superior to the Auditory modality in every category except physical demand. The lack of significance for physical demand is more than likely due to the evaluation requiring very little physical effort by the participant for all of the modalities tested. Comparing NASA TLX ratings for the Communications Mode and Paper modality revealed that the Communications Mode produced lower workload than the Paper modality in every category except physical demand and performance. The lack of significance for the physical demand comparison for the Communications Mode and Paper modality is likely attributed to the fact that all three of the modalities required very little physical effort to use. The lack of significance for the performance category is likely due to the similarities between the Communications Mode and Paper modality. In both, performance was highly rated, thus supporting the hypothesis that the Communications Mode performs at least as well as the Paper modality, if not better.

A goal of the evaluation was to determine the most appropriate means of presenting task-specific information for mHRI tasks. The results of this evaluation indicate that visual-based modalities (such as the Communications Mode and Paper) are better suited to mHRI tasks using the MMBTI than auditory-based modalities. These results also show that the Paper modality provides no usability benefit versus the Communications Mode. Therefore, since the Communications Mode is integrated into the MMBTI (thus requiring no external sources to be fully utilized) it is the ideal choice for mHRI tasks amongst the three modalities evaluated.

CHAPTER V

Reduction Methods Evaluation

V.1 Introduction

A participant evaluation was performed in order to assess the benefits of the developed reduction methods. The primary purpose of the Reduction Methods Evaluation was to determine the best method for reducing a participant's geospatial input into the MMBTI. In addition to this primary goal, two other hypotheses were investigated. First, it was hypothesized that participants would supply more erroneous input when using touch-based interaction versus mouse-based interaction. It was also hypothesized that participants would provide input at a higher rate (i.e., participants would draw lines and areas with a higher velocity) when using touch-based interaction.

V.2 Method

The Reduction Methods Evaluation was performed using a Dell XT2 tablet laptop running Windows© 7. Logged data included: the error associated with the participant's input for the shape the participant was asked to draw, drawing duration, and the error associated with each interpolation method as compared to participants' actual input.

Participants received a five-minute period of time to familiarize themselves with the test application. If desired, the participant drew a number of lines in the application in order to understand how the application worked. After the participant was comfortable using the test application, the evaluation was conducted.

A with-in subjects evaluation was designed to evaluate both input methods (i.e., touch-based versus mouse-based) and the data reduction methods. Presentation of the input method condition was randomized amongst participants. Since the study design was with-in subjects, each participant was required to trace the twelve shapes with both touch-based and mouse-based interaction. Presentation of the shapes was randomized across conditions (i.e., touch-based and mouse-based interaction), but the same across participants. The independent variable was the input method. Dependent variables were the participant's drawing speed, error rates of the data reduction methods, and the error between the participant's supplied input and the shape being traced. No subjective evaluations were performed.

Error rates and drawing speed were used as objective measures to evaluate the input condition. Error rates were also used as objective measures to evaluate the efficacy of the data reduction methods. Error was calculated using the methods described in Chapter III.2. Drawing Speed was calculated by measuring the elapsed time between when a participant began drawing a shape until the participant finished drawing the

shape. This length of the drawn line (measured in pixels) was then divided by the elapsed time to provide velocity as measured in pixels per second ($\frac{px}{sec}$).

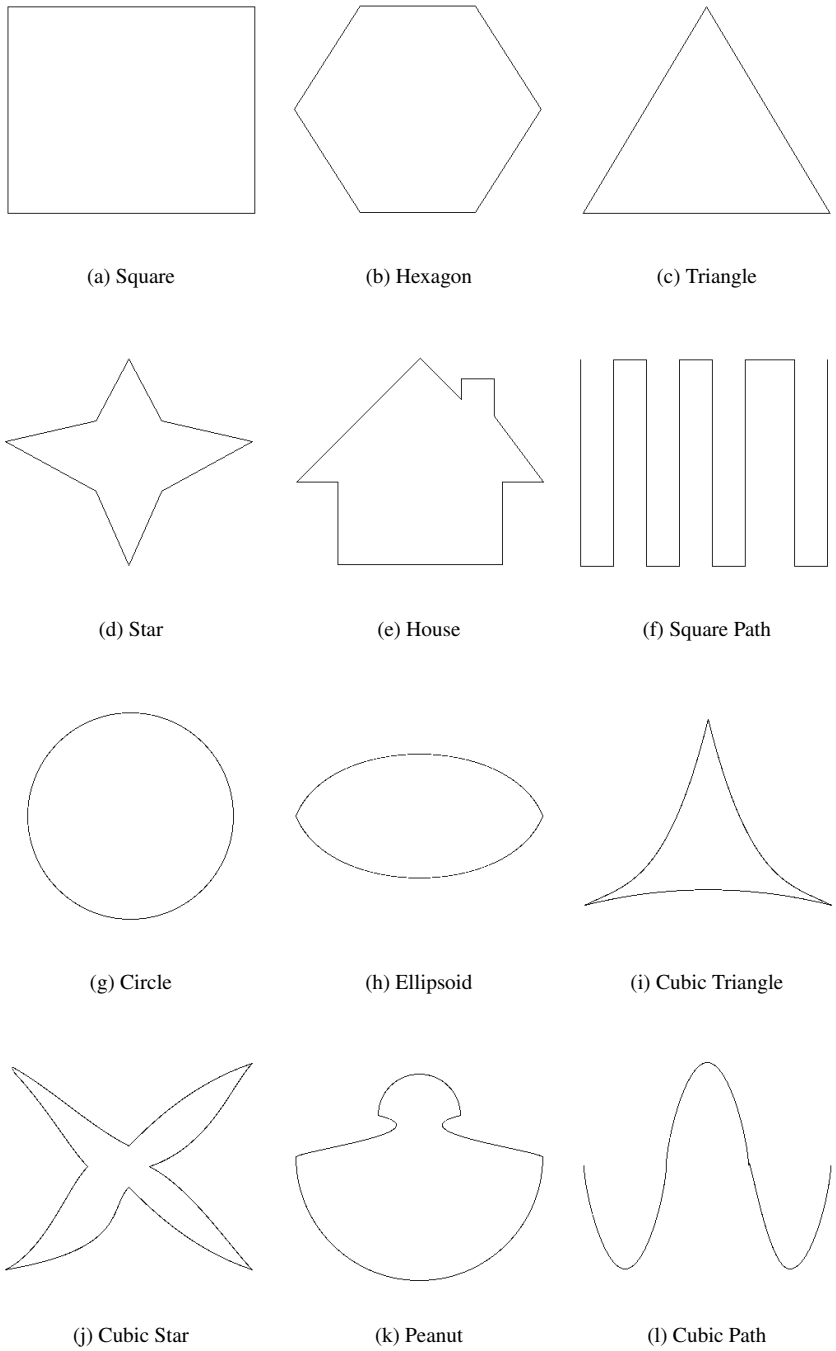
Ten participants performed the evaluation. Six were male and four were female. Two participants were left-handed and eight were right-handed, although all ten used the mouse with their right hand. The average age of the participants was 23.8 ± 3.79 years. Six participants reported a good understanding of mouse-based drawing programs (such as Microsoft© Paint), while four reported a basic understanding of such programs. Five participants reported no experience with touch-based drawing programs, four reported a basic understanding, and only one reported a good understanding of such programs.

V.3 Test Application Description

A test application was developed to perform the Reduction Methods Evaluation. The test application was a full-screen application that displayed shapes using a one pixel wide black line on a white background. Shapes were displayed one at a time, and participants were required to draw each shape. Twelve shapes were tested and each shape fell into one of two categories: linear or curved. Linear shapes were comprised entirely of straight lines and curved shapes were constructed using cubic Bézier segments. Shapes in each category varied by the number of segments that comprised the shape. In order to keep linear and curved shapes similar, shapes in each category were designed using an equal number of line segments, whenever possible. For example, if a linear shape was comprised of four segments, there also existed a curved shape that was comprised of four segments. All twelve shapes were roughly the same size and were limited to a region of 300 *x* 250 pixels. Each of the twelve shapes is shown in reduced scale in Figure V.1.

As the participant supplied input, the online data reduction algorithms reduced the data and the reduced data set error was calculated per shape for each method. These error calculations were stored. Error was also calculated and stored for the participant's drawn line versus the shape the participant was attempting to draw. An error calculation for participant input compared to the drawn line was performed after the participant finished drawing the shape; therefore, participant performance was not affected by any slowdown due to computational overhead. The test application allowed participants to see their input data as they were drawing (similar to how a line is shown when drawing in a paint program), but the reduced data sets for each method were not displayed.

At test application startup, the participant selected either the Mouse or Touch push button to indicate which input method he/she intended to use. The participant pressed the Next Shape button to see the first shape to be drawn. Users traced the displayed shape using one continuous line. Once the participant finished tracing the shape and the drawing error was calculated, the Next Shape button became selectable and the participant pressed it to see the next shape. This process was repeated until all twelve shapes were drawn.



(a) Square (b) Hexagon (c) Triangle

(d) Star (e) House (f) Square Path

(g) Circle (h) Ellipsoid (i) Cubic Triangle

(j) Cubic Star (k) Peanut (l) Cubic Path

Figure V.1: The shapes tested. Shapes (a) through (f) are linear, while shapes (g) through (l) are curved.

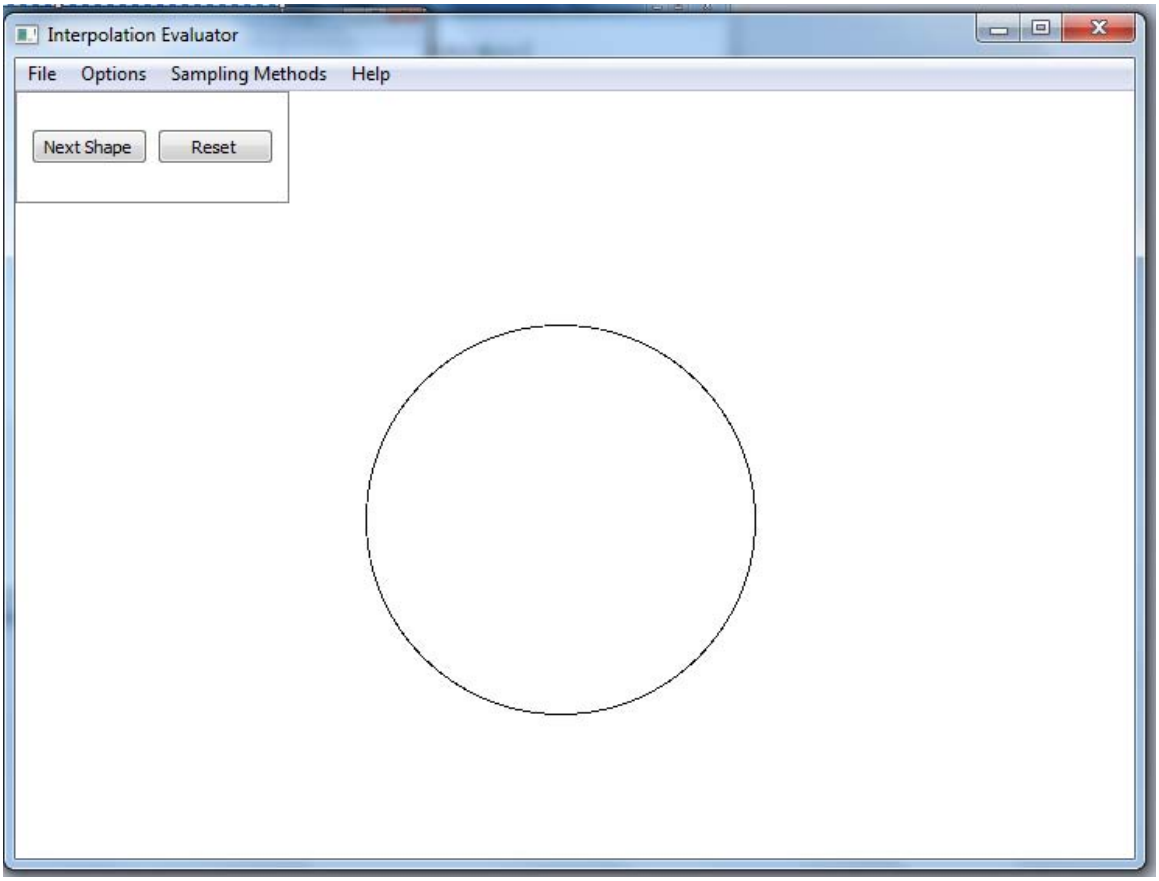


Figure V.2: The Reduction Methods Evaluator application

Once the participant had drawn all twelve shapes using either the touch or the mouse, the participant repeated the process using the alternate input method. The shapes were shown in a different order depending on which input method was selected in order to minimize learning effects, but all twelve shapes were used with each input method. A screenshot of the test application can be seen in Figure V.2.

V.4 User Input Method Results

V.4.1 Mouse-Based and Touch-Based Input Comparison

Error was measured between the participant's input data set and the shape the participant was attempting to trace in order to objectively compare mouse and touch-based interaction for drawing. The error rates for each shape were calculated and averaged per point to determine the error per point for each point in the participant's input data set. The error rates for each participant were summed according to shape type and input type in order to provide an error score. Error scores are summarized in Table V.1.

A multivariate test using ANOVA was performed in order to analyze the significance of the main effects (input method and shape type) for this evaluation. The multivariate test determined with significance that the

	Mouse Error Score		Touch Error Score	
Participant	**Linear**	**Curved**	**Linear**	**Curved**
1	15.538	16.267	29.081	38.063
2	8.953	11.010	17.270	20.819
3	10.687	12.923	26.644	28.692
4	8.644	10.229	21.498	19.280
5	6.998	8.226	19.284	22.936
6	7.506	9.664	19.110	19.084
7	10.817	14.014	23.023	20.765
8	15.570	17.122	20.549	19.497
9	16.009	14.661	22.490	22.441
10	7.907	8.930	22.682	20.736
Mean	10.863	12.305	22.163	23.231
Std	3.560	3.140	3.554	5.920

Table V.1: Error scores per participant by shape type and input method. Error scores are the summed average of error rates per shape.

	Error Score	
Participant	**Touch**	**Mouse**
1	33.572	15.902
2	19.045	9.982
3	27.668	11.805
4	20.389	9.437
5	21.110	7.612
6	19.097	8.585
7	21.894	12.415
8	20.023	16.346
9	22.465	15.335
10	21.709	8.418
Mean	22.697	11.584
Std	4.545	3.302

Table V.2: Error scores reported by input method.

input condition was a main effect ($F_{1,9} = 71.8, p \leq 0.001$). A multivariate test to determine if the shape type was a main effect was nearly significant ($F_{1,9} = 4.98, p = 0.052$). Therefore, a t-test contrast on shape type was performed as well.

Since the input method was determined to be a main effect, mouse-based interaction and touch-based interaction were contrasted. The null hypothesis was that touch-based interaction results in less error than mouse-based interaction. Error data was categorized by input method only (see Table V.2), and a paired, two-tailed t-test using the data in Table V.2 was performed. The mean of the error score differences for the input method was significant ($t_{19} = -11.11, p \leq 0.0001$), refuting the null hypothesis. Therefore, touch-based interaction results in more error than mouse-based interaction. Error scores based on input method for each participant is summarized in Figure V.3.

The shape type was also observed to be a main effect with near significance ($F_{1,9} = 4.98, p = 0.052$). The null hypothesis was made that linear shapes result in higher error than curved shapes regardless of input method. Error data was categorized by shape type (see Table V.3) and subjected to a paired, two-tailed t-test to evaluate the null hypothesis. The mean of the error score differences for shape type was significant

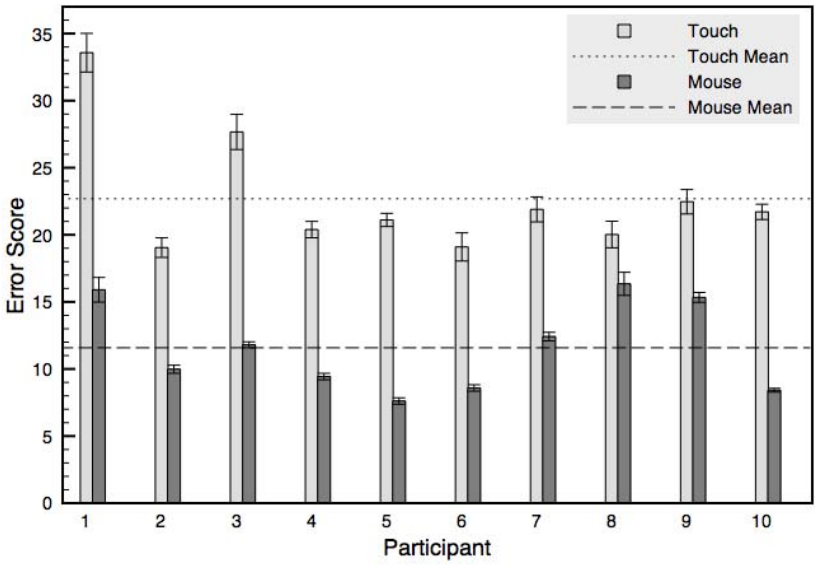


Figure V.3: Averaged error scores for each participant shown by input type.

	Error Score	
Participant	**Linear**	**Curved**
1	22.310	27.165
2	13.112	15.915
3	18.665	20.808
4	15.071	14.755
5	13.141	15.581
6	13.308	14.374
7	16.920	17.389
8	18.060	18.310
9	19.249	18.551
10	15.294	14.833
Mean	16.513	17.768
Std	3.074	3.888

Table V.3: Error scores reported by shape type.

($t_{19} = 20, p \leq 0.05$), refuting the null hypothesis. Therefore, drawing curved shapes results in higher error than linear shapes regardless of the input method used.

The shape type was also considered for each of the input methods individually and error scores can be seen in Table V.1. The null hypothesis was that curved shapes result in less error than linear shapes for mouse-based interaction. A paired, two-tailed t-test was performed to confirm the null hypothesis. The mean of the error score differences was significant ($t_9 = -4.17, p \leq 0.005$), refuting the null hypothesis and proving that linear shapes result in less error than curved shapes for mouse-based interaction.

The null hypothesis was made that curved shapes would result in less error than linear shapes for touch based interaction. A paired, two-tailed t-test was performed in order to confirm the null hypothesis. The mean of the error score differences was not significant in this case ($t_9 = -1.048, p = 0.32$). Therefore, it

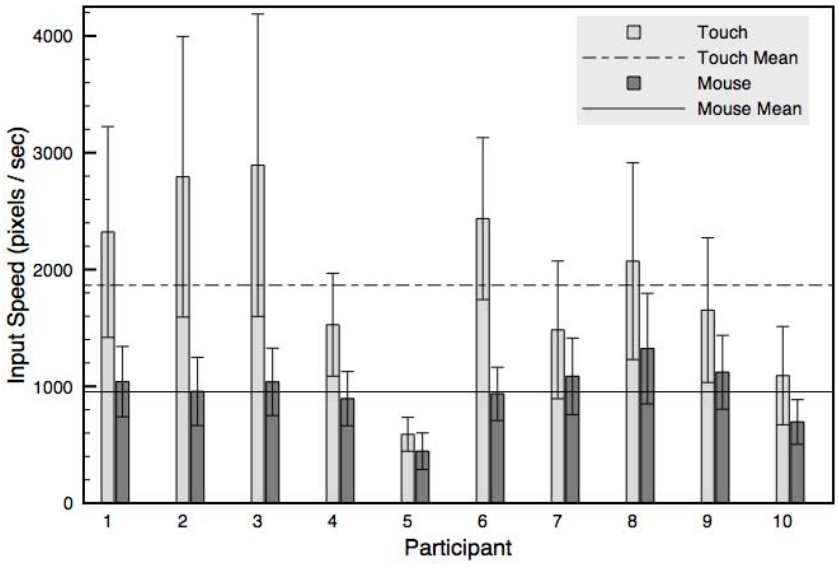


Figure V.4: Average drawing speed for each participant using mouse-based and touch-based interaction.

cannot be determined if curved shapes or linear shapes result in more error when drawing using touch-based interaction.

Drawing speed using each of the input methods was also considered. The significance of the main effects on drawing speed were determined via a multivariate analysis using ANOVA. It was determined that the input method main effect for drawing speed was significant ($F_{1,9} = 21.80, p \leq 0.002$). The main effect of shape type was not a significant main effect ($F_{1,9} = 0.278, p \leq 0.611$). Recorded average drawing speeds (measured in $\frac{px}{sec}$) can be seen in Figure V.4 .

Since the input method was determined to be a significant main effect, a two-tailed, paired t-test was conducted to test the null hypothesis that mouse-based interaction is faster than touch-based interaction. The mean of the drawing speed differences for mouse-based input was significant ($t_{19} = -6.54, p \leq 0.001$), refuting the null hypothesis. Therefore, the mean drawing speed is faster for touch-based interaction.

V.4.2 Discussion

The comparisons between mouse and touch-based input appear to support the claims of Muratore (1987); Ahlström and Lenman (1987); and Karat et al. (1986) which state that touch-based interaction is faster, but less accurate, than mouse-based interaction. Using a quantifiable error metric, it was determined that touch-based interaction results in more error than mouse-based interaction for drawing tasks of either linear or curved shape types. It was also determined that drawing curved shapes results in more error than drawing linear shapes for mouse-based interaction; however, it cannot be determined if drawing linear shapes or

curved shapes resulted in higher error in touch-based interaction. It is expected that with more participants, a significant result can be obtained.

The drawing speed of touch-based interaction versus mouse-based interaction were also compared. Once again, the claimes of Muratore (1987); Ahlström and Lenman (1987); and Karat et al. (1986) were supported. It was determined that the input method was the only main effect on drawing speed for this evaluation. It was also shown that touch-based interaction is significantly faster than mouse-based interaction for drawing tasks.

The goal was to determine the deficiencies of touch-based interaction versus mouse-based interaction. Results seem to indicate that higher error rates using touch-based interaction are unavoidable, and means should be provided to support the participant during drawing tasks. Therefore, future designs should focus on methods within the interface to improve touch-based interaction such that error rates for touch-based drawing tasks can be reduced.

V.4.3 Reduction Methods Results

The overall best reduction method is the method that provides the greatest amount of reduction and has a low error score. Therefore, the error score for each method and the size of the reduced data set for each method will be compared. Pairwise comparisons were made using all possible combinations of input methods to determine which reduction method produced the lowest error score. Rather than conduct individual pairwise t-tests, the Tukey post-hoc test was used. Since six pairwise comparisons must be made, it is likely that a significant result can be obtained by chance. Therefore, the Tukey test was used in order to prevent false significance. Four different tests were performed in order to determine the efficacy of the reduction methods. These tests included using the mouse to draw linear shapes (Mouse-Linear), using the mouse to draw curved shapes (Mouse-Curved), using touch to draw linear shapes (Touch-Linear), and using touch to draw curved shapes (Touch-Curved). Error scores of the four reduction methods for each condition can be seen in Figure V.5.

The error scores for the Mouse-Linear condition can be seen in Table V.4. The Tukey test was used to determine the significant comparisons amongst the six possible comparisons. The results of the Tukey test can be seen in Table V.5.

Tukey post-hoc comparison results are reported in terms of a 95% Confidence Interval (CI) and a p-value. The Tukey post-hoc comparisons of the four methods indicate that the Cubic method had significantly lower error than the Slope method (95% CI $[-22.268, 7.211], p = 0.002$) and the Iterative method (95% CI $[261.179, 276.237], p \leq 0.001$). The Iterative Cubic ($\mu = 0.409, Std = 0.015$, 95% CI $[2.941, 33.056]$) method resulted significantly lower error when compared to the Slope method ($p = 0.014$). The Iterative Cubic

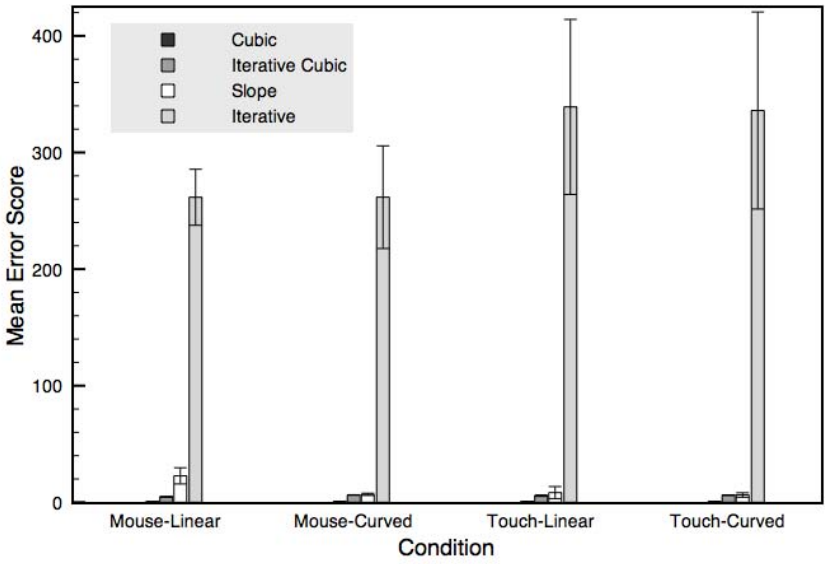


Figure V.5: Average error scores of the four reduction methods for each input condition.

Participant	Cubic Error	Iterative Cubic Error	Slope Error	Iterative Error
1	0.406	4.747	17.703	256.964
2	0.396	4.352	33.539	259.924
3	0.401	4.627	19.918	262.973
4	0.404	4.399	27.405	245.442
5	0.414	4.627	24.908	227.296
6	0.388	4.362	31.521	248.658
7	0.430	5.038	24.838	284.311
8	0.436	5.139	11.528	316.382
9	0.406	4.747	17.703	256.964
10	0.406	4.747	17.703	256.964
Mean	0.409	4.678	22.677	261.588
Std	0.015	0.268	6.947	24.017

Table V.4: Error scores for each of the reduction methods for the Mouse-Linear condition.

		95% Confidence Interval for Mean		
Method	**Mean Difference**	**Lower Bound**	**Upper Bound**	**p adjusted**
Iterative Cubic - Cubic	4.270	-10.788	19.327	0.870
Slope - Cubic	22.268	7.211	37.326	0.002
Iterative - Cubic	261.179	246.122	276.237	≤ 0.001
Slope - Iterative Cubic	17.998	2.941	33.056	0.014
Iterative - Iterative Cubic	256.909	241.852	271.967	≤ 0.001
Iterative - Slope	238.911	223.854	253.969	≤ 0.001

Table V.5: Results of the Tukey test for the Mouse-Linear condition. Note that all comparisons except Iterative Cubic - Cubic are significant for the Mouse-Linear condition.

Participant	Cubic Error	Iterative Cubic Error	Slope Error	Iterative Error
1	0.525	6.024	6.095	351.633
2	0.587	6.231	7.116	243.656
3	0.563	6.109	7.344	237.530
4	0.581	6.314	7.661	233.866
5	0.586	6.420	8.444	220.067
6	0.605	6.330	7.526	234.087
7	0.556	6.201	5.593	289.148
8	0.600	6.699	5.582	322.204
9	0.588	6.277	6.526	242.180
10	0.588	6.277	6.526	242.180
Mean	0.578	6.288	6.841	261.655
Std	0.024	0.183	0.939	44.026

Table V.6: Error scores for each of the reduction methods for the Mouse-Curved condition.

		95% Confidence Interval for Mean		
Method	Mean Difference	Lower Bound	Upper Bound	p adjusted
Iterative Cubic - Cubic	5.710	-20.809	32.230	0.937
Slope - Cubic	6.264	-20.256	32.783	0.920
Iterative - Cubic	261.077	234.557	287.597	≤ 0.001
Slope - Iterative Cubic	0.553	-25.966	27.073	1.000
Iterative - Iterative Cubic	255.367	228.847	281.887	≤ 0.001
Iterative - Slope	254.814	228.294	281.333	≤ 0.001

Table V.7: Results of the Tukey test for the Mouse-Curved condition. Note that the Iterative - Cubic, Iterative - Iterative Cubic, and Iterative-Slope comparisons are significant .

method also resulted in significantly lower error than the Iterative method (95% CI $[241.852, 271.967], p \leq 0.001$). The Slope method also resulted in significantly lower error than the Iterative method (95% CI $[223.854, 253.969], p \leq 0.001$).

Based on the findings from the Mouse-Linear test, it appears that curve methods are superior to linear methods for reducing data sets supplied through mouse-based interaction that represent linear shapes.

The error scores for the Mouse-Curved condition can be seen in Table V.6. The Tukey test was used to determine the significant comparisons amongst the six possible comparisons. The results of the Tukey test can be seen in Table V.7.

The Tukey post-hoc comparisons for the four methods indicate that the Cubic method resulted significantly lower error than the Iterative method (95% CI $[234.557, 287.597], p \leq 0.001$). The Iterative Cubic method also had significantly less error than the Iterative method (95% CI $[228.847, 281.887], p \leq 0.001$). The Slope method also resulted in significantly less error than the Iterative method (95% CI $[228.294, 281.333]$, $p \leq 0.001$).

Based on these findings, it appears that of the linear methods, Slope reduction results in less error than Iterative reduction for mouse-based interaction. Of the curve methods, it is inconclusive which method results in less error.

The error scores for the Touch-Linear condition can be seen in Table V.8. The Tukey test was used to

Participant	Cubic Error	Iterative Cubic Error	Slope Error	Iterative Error
1	0.500	5.430	5.794	392.173
2	0.522	6.199	5.003	426.315
3	0.537	6.232	4.251	429.459
4	0.494	5.482	6.380	307.756
5	0.518	5.635	9.739	232.804
6	0.501	6.258	4.746	398.838
7	0.491	5.704	6.454	307.960
8	0.528	6.029	5.476	381.513
9	0.406	4.747	17.703	256.964
10	0.406	4.747	17.703	256.964
Mean	0.490	5.646	8.325	339.075
Std	0.047	0.564	5.166	75.055

Table V.8: Error scores for each of the reduction methods for the Touch-Linear condition.

		95% Confidence Interval for Mean		
Method	Mean Difference	Lower Bound	Upper Bound	p adjusted
Iterative Cubic - Cubic	5.156	-40.152	50.464	0.990
Slope - Cubic	7.835	-37.473	53.143	0.966
Iterative - Cubic	338.584	293.276	383.892	≤ 0.001
Slope - Iterative Cubic	2.679	-42.629	47.987	0.999
Iterative - Iterative Cubic	333.428	288.120	378.736	≤ 0.001
Iterative - Slope	330.750	285.442	376.058	≤ 0.001

Table V.9: Results of the Tukey test for the Touch-Linear condition. Note that all comparisons against the Iterative method are significant.

determine the significant comparisons amongst the six possible comparisons. The results of the Tukey test can be seen in Table V.9.

The Tukey post-hoc comparisons for the Touch-Linear condition indicate that the Cubic method resulted in significantly lower error than the Iterative method (95% CI $[293.276, 383.892], p \leq 0.001$). The Iterative Cubic method also had significantly lower error than the Iterative method (95% CI $[288.120, 378.736], p \leq 0.001$). The Slope method also had significantly lower error than the Iterative method (95% CI $[285.442, 376.058]$, $p \leq 0.001$).

These results indicate that the Iterative method is inferior to the other methods for reducing touch-based input on linear shapes. Comparisons between the other three methods were insignificant; therefore, it is unclear which is the best for reducing touch-based input on linear shapes.

The error scores for the Touch-Curved condition can be seen in Table V.10. The Tukey test was used to determine the significant comparisons amongst the six possible comparisons. The results of the Tukey test can be seen in Table V.11.

Tukey post-hoc comparisons for the Touch-Curved condition indicate that the Cubic method resulted in significantly lower error than the Iterative method (95% CI $[284.567, 386.367], p \leq 0.001$). The Iterative Cubic method also had significantly lower error than the Iterative method (95% CI $[278.951, 380.751], p \leq 0.001$). The Slope method also resulted in significantly lower error than the Iterative method (95% CI

Participant	Cubic Error	Iterative Cubic Error	Slope Error	Iterative Error
1	0.525	6.024	6.095	351.633
2	0.531	6.354	4.136	447.111
3	0.530	6.304	4.080	434.855
4	0.498	5.830	6.365	317.783
5	0.538	5.697	10.605	218.625
6	0.506	6.481	4.782	430.938
7	0.527	6.023	7.670	313.195
8	0.512	6.236	5.864	361.515
9	0.588	6.277	6.526	242.180
10	0.588	6.277	6.526	242.180
Mean	0.535	6.150	6.265	336.001
Std	0.031	0.248	1.905	84.498

Table V.10: Error scores for each of the reduction methods for the Touch-Curved condition.

		95% Confidence Interval for Mean		
Method	Mean Difference	Lower Bound	Upper Bound	p adjusted
Iterative Cubic - Cubic	5.616	-45.284	56.516	0.991
Slope - Cubic	5.730	-45.170	56.630	0.990
Iterative - Cubic	335.467	284.567	386.367	≤ 0.001
Slope - Iterative Cubic	0.115	-50.785	51.015	1.000
Iterative - Iterative Cubic	329.851	278.951	380.751	≤ 0.001
Iterative - Slope	329.736	278.836	380.636	≤ 0.001

Table V.11: Results of the Tukey test for the Touch-Curved condition. Note that the Iterative-Cubic, Iterative-Iterative Cubic, and Iterative-Slope comparisons are significant.

$[278.836, 380.636], p \leq 0.001$).

These results, along with those from the Touch-Linear condition, indicate that Iterative reduction is inferior to the three other methods for reduction touch-based input for drawing tasks. The lack of significant results from all other pairwise comparisons indicate that it is inconclusive as to which reduction method is the best for touch-based interaction amongst the Slope, Cubic, and Iterative Cubic methods.

A comparison of each method's reduction ability was conducted in order to shed more insight on the effectiveness of the methods. Reduction results arranged by input method and shape type can be seen in Tables V.12 and V.13, respectively. Figure V.6 displays the average reduced data set size using each reduction method and the mean unreduced data set sizes for touch-based and mouse-based interaction.

It is not surprising that Iterative reduction has the largest mean reduction value in Tables V.12 and V.13. The large error rates reported for Iterative reduction, seen in Tables V.4, V.6, V.8, V.10 indicated that Iterative reduction could be over-reducing the input data set, thus causing a very large amount of error for all conditions tested.

Very little reduction was seen in the case of Cubic reduction for both the input method and shape conditions. This lack of reduction was expected and explains why Cubic reduction resulted in very low error scores in Tables V.12 and V.13.

A two-tailed, paired t-test determined that Iterative Cubic reduction provided significantly more reduc-

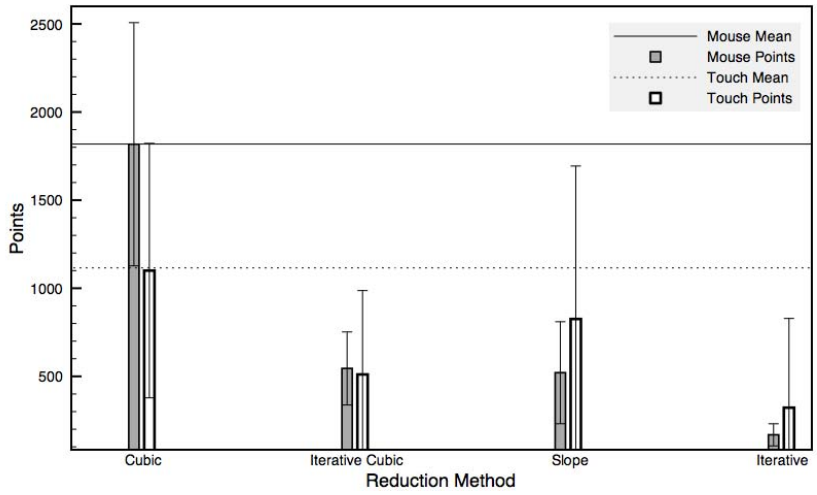


Figure V.6: Average set sizes for each reduction method for both mouse-based and touch-based interaction.

Input Method		Mouse	Touch
Input	**Mean**	1818.257	1115.463
	Std	690.074	278.863
	Reduction	0.000	0.000
Cubic	**Mean**	1817.228	1100.233
	Std	690.046	722.121
	Reduction	1.030	15.230
Iterative Cubic	**Mean**	544.878	510.843
	Std	207.263	476.020
	Reduction	1273.380	604.620
Slope	**Mean**	520.674	825.524
	Std	289.411	868.121
	Reduction	1297.583	289.939
Iterative	**Mean**	168.153	322.317
	Std	62.748	506.487
	Reduction	1650.105	793.146

Table V.12: Mean sizes of the reduced data sets for each reduction method and the input data set categorized by input method. The Reduction value is the difference in set sizes for the reduction method versus the input set.

tion than Cubic reduction ($t_{15} = -2.82, p \leq 0.002$), when the null hypothesis was that Cubic reduction provides greater reduction than Iterative Cubic reduction. A null hypothesis was that Cubic reduction results in significantly greater reduction than Slope reduction. A two-tailed, paired t-test rejected the null hypothesis and determined that Slope reduction results in significantly greater reduction than Cubic reduction ($t_{15} = -1.72, p \leq 0.002$). The lack of significance in error scores between the Cubic-Iterative Cubic comparison in Tables V.4, V.6, V.8, and V.10 indicates that there is not a significant error difference between Cubic reduction, which provides very little to no reduction, and the Cubic Iterative method, which provides a great deal more reduction than the Cubic method.

The Slope reduction method appears to provide slightly better average reduction versus Iterative Cubic reduction for the case of mouse-based interaction and for linear shape types (see Tables V.12 and V.13).

Input Method		Linear	Curved
Input	**Mean**	1505.644	1428.076
	Std	726.160	541.098
	Reduction	0.000	0.000
Cubic	**Mean**	1490.841	1426.620
	Std	735.995	543.489
	Reduction	14.803	1.456
Iterative Cubic	**Mean**	591.843	458.001
	Std	458.332	153.099
	Reduction	913.801	970.075
Slope	**Mean**	566.089	638.568
	Std	701.194	304.485
	Reduction	939.555	789.508
Iterative	**Mean**	242.422	186.369
	Std	390.052	129.654
	Reduction	1263.222	1241.707

Table V.13: Mean sizes of the reduced data sets for each reduction method and the input data set categorized by shape type. The Reduction value is the difference in set sizes for the reduction method versus the input set.

A null hypothesis was that Slope reduction resulted in greater reduction than Iterative Cubic reduction for touch-based interaction. However, a two-tailed, paired t-test determined that Iterative Cubic reduction was significantly better ($t_{15} = -4.36, p \leq 0.006$) than Slope reduction for touch-based interaction.

V.4.4 Reduction Methods Discussion

Based on the error score findings from the Reduction Methods Evaluation, no definitive method can be pointed to as best for touch-based (or even mouse-based) line drawing. As a result of the pairwise comparisons in the Mouse-Linear, Mouse-Curved, Touch-Linear, Touch-Curved conditions; it is obvious that the Iterative method is not ideal for reducing participant input. However, it is expected that the poor performance of Iterative reduction is due to the k reduction index used to reduce the input data set. If anything, data indicates that k should be smaller than the value of ten used in this evaluation.

When comparing reduction, it was determined that the Iterative method, due to its value for k, resulted in the most reduction of the input data set for all cases. However, due to Iterative reduction's high error at its current k value, this method is not considered a viable option for reducing a participant's input data set. Findings showed that Cubic reduction resulted in the least amount of reduction, this finding was expected based on the error values reported for Cubic reduction. Iterative-Cubic reduction and Slope reduction provided significantly more reduction than Cubic reduction for both touch and mouse-based interaction, while generating error scores that were not significantly higher than Cubic reduction. However, in the case of touch-based interaction, the condition of particular interest for the MMBTI, Iterative Cubic reduction proved to be significantly greater than Slope reduction. Therefore, for touch-based interaction, it appears that Iterative Cubic reduction may be a valid method for reducing participant input data sets. It is believed that with additional

experimentation, the Iterative Cubic method may be further improved by implementing a more intelligent method of determining the appropriate control points for a data set reduced using the method.

Perhaps the most worthwhile finding from this work was the development of an error calculation method that is guaranteed to converge. Since the developed error method has guaranteed convergence, it can perhaps become a component of a new reduction method. This new reduction method can reduce the data set by some metric, sample the error, and attempt to reduce the line again with a different method in an attempt to reduce the calculated error. It is suspected that for a reduction method that utilizes error detection to be successful, the efficiency of the error method will have to be improved.

It should also be noted that the reduction methods provided in this thesis are relatively simplistic in their design. The benefit to their design is that they can be easily implemented in real time with no noticeable overhead. However, more adequate solutions can be pursued. Based on the results of this evaluation, it seems worthwhile to pursue a method based on Iterative Cubic reduction that attempts to leverage the error metric in order to best reduce a participant's input data set with minimal error incurred.

V.5 Conclusion

A study of participant input methods determined that, for drawing tasks, touch-based interaction was faster, but less accurate than mouse-based interaction. It is important to consider the design ramifications of this finding on the MMBTI. As design of the MMBTI moves forward, special attention will have to be paid to touch-based input, especially for drawing tasks.

A study of methods for reducing a user's input data during drawing tasks was also conducted. Findings from this study determined that the Iterative Cubic reduction method may be the best method of the four presented. However, each of the four methods was simplistic, and an appropriate solution will not be found until more sophisticated reduction methods are developed and studied. Moving forward, the Iterative Cubic reduction method seems to provide an adequate starting point for developing a more sophisticated method. It is suspected that an ideal reduction method will be capable of error detection and correction. The combination of the error detection method developed for this thesis with the Iterative Cubic method can result in a more robust reduction method.

CHAPTER VI

Concluding Remarks

VI.1 Introduction

This thesis discussed the development and implementation of a Communications Mode for the MMBTI and methods of reducing a user's input data set. The main contribution of this thesis was to develop methods to facilitate the touch-based input of drawing tasks within the MMBTI. The Communications Mode was developed in order to provide an effective means of conveying task-based information to mobile users during mHRI tasks. The integration of the Communications Mode into the MMBTI results in a mobile interface capable of supporting a user performing mHRI tasks. The data reduction methods served to shed insight on the problem of reducing a user's geospatial input data set for the MMBTI. A method of solving the nearest-point-on-a-curve method with guaranteed convergence is also a contribution of this work.

VI.2 The Communications Mode

A user evaluation determined that the Communications Mode was superior to the Auditory modality and comparable, if not superior to, the Paper modality. It is important to note that superiority versus the Paper modality is not required. As long as the Communications Mode introduces no deficiencies versus the Paper modality, it can be used for mHRI tasks, since it requires no external sources to convey task information. With the integration of the Communications Mode into the MMBTI, the MMBTI became a suitable interface for mHRI tasks, which was the Communications Mode's main contribution. The Communications Mode evaluation also determined that digital maps can be as good as, if not superior to, paper-based maps when provided with additional on-screen information, which was also a contribution.

The Communications Mode serves as a major design milestone for the MMBTI. With its implementation, the MMBTI can be tested in a mobile context to determine the interface's applicability to mHRI, which is the ultimate goal. The Communications Mode also provides a robust method for communicating information to a mobile user. Now that the Communications Mode is in place, additional features can be added and tested to determine if they are a benefit to mHRI.

VI.3 Reduction Methods

The developed reduction methods were simplistic, but nonetheless shed insight on the problem of reducing a user's geospatial input into the MMBTI. The Iterative Cubic method was shown to hold the most promise for high levels of reduction with minimal error; however, the method could be further developed to increase its

effectiveness. It is believed that by implementing a better method for control point selection in Algorithm 3, the Iterative Cubic method can be improved.

As a result of the reduction methods evaluation, fundamental findings concerning mouse-based and touch-based interaction were also determined. A contribution of this work was the confirmation of previous findings by Muratore (1987); Ahlström and Lenman (1987); and Karat et al. (1986) which stated that touch-based interaction results in more error, but faster interaction times. The reduction methods evaluation confirmed this finding for drawing tasks. The use of a mouse is unacceptable in mobile interaction, therefore interface designers must develop and implement methods to leverage the benefits of touch-based drawing (i.e., drawing speed), while compensating for its deficiencies (i.e., error detection and correction).

Another contribution of this thesis is the development of an error detection method that is guaranteed to converge. This error detection method, once optimized, can be applied as a means of real-time error detection for touch-based drawing tasks. Once the error method can detect errors efficiently in real time, it can serve as the input to an error correction algorithm that attempts to reduce the calculated error. It is important to note that, in order to be successfully integrated into the MMBTI (a much larger and more complex system), the optimization of the error detection method is crucial.

VI.4 Future Work

The Communications Mode serves as a robust mechanism for implementing and testing communicative features in the MMBTI. Therefore, a multitude of design ideas can now be developed, implemented, and tested. However, it is crucial to determine the applicability of the Communications Mode to mHRI. Therefore, a mobile study will be conducted using the Communications Mode to determine if any modifications or improvements need to be made to make it suitable for mHRI tasks.

The reduction methods served as an initial attempt to reduce a user's input data set during drawing tasks. Therefore, there is a large amount of room for improvement. Based on the results of the reduction methods evaluation, it seems as though the best starting point for a more sophisticated reduction method is Iterative Cubic reduction. Initial work will be to make Iterative Cubic's control point calculation more sophisticated. Iterative Cubic reduction can also be combined with the error detection method introduced in this thesis to provide a method that facilitates rapid error detection and correction. Such an approach is ideal, but only if it is optimized to the point that it can work in real time with no noticeable overhead. Once a sophisticated reduction method is developed, it will also have to be implemented into the MMBTI.

BIBLIOGRAPHY

Ahlström, B. and Lenman, S. (1987). Touch screen, cursor keys and mouse interaction. In *Selected papers from the International Scientific Conference on Work with display units 86*, pages 831–837, Amsterdam, The Netherlands, The Netherlands. North-Holland Publishing Co.

Albinsson, P. and Zhai, S. (2003). High precision touch screen interaction. In *Proceedings of the conference on Human factors in computing systems - CHI '03*, page 105, Ft. Lauderdale, Florida, USA.

Apple (2010a). Apple - iPad - see the web, email, and photos like never before. http://www.apple.com/ipad/.

Apple (2010b). Apple - iPhone 4 - video calls, multitasking, HD video, and more. http://www.apple.com/iphone/.

Bederson, B. B. and Hollun, J. D. (1994). Pad++: A zooming graphical interface for exploring alternate interface physics. In *In Proceedings of User Interface and Software Technology*.

Benko, H., Wilson, A. D., and Baudisch, P. (2006). Precise selection techniques for Multi-Touch screens. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 1263–1272.

Burnette, E. (2010). How to use multi-touch in android 2 | ZDNet. http://www.zdnet.com/blog/burnette/how-to-use-multi-touch-in-android-2/1747?tag=mantle_skin;content.

Cacciabue, P. C. and Martinetto, M. (2006). A user-centred approach for designing driving support systems: the case of collision avoidance. *Cognition, Technology & Work*, 8(3):201–214.

Chittaro, L. (2006). Visualizing information on mobile devices. *Computer*, 39(3):40–45.

Cohen, P., McGee, D., and Clow, J. (2000). The efficiency of multimodal interaction for a map-based task. In *Proceedings of the sixth conference on Applied natural language processing*, pages 331–338, Seattle, Washington. Association for Computational Linguistics.

Cohen, P. R., Johnston, M., McGee, D., Oviat, S. L., Clow, J., and Smith, I. (1998). The efficiency of multimodal interaction: A case study. In *Proceedings of the International Conference on Spoken Language Processing*, pages 249–252, Sydney.

Davies, C. (2009). Qt 4.6 with multitouch released plus second maemo 5 tech preview [Video] - SlashGear. http://www.slashgear.com/qt-4-6-with-multitouch-released-plus-second-maemo-5-tech-preview-video-0164775/.

Dell (2010). Latitude XT2 tablet | dell. http://www.dell.com/tablet.

Dowell, J. and Shmueli, Y. (2008). Blending speech output and visual text in the multimodal interface. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 50(5):782 –788.

Forlines, C. and Shen, C. (2005). DTLens: multi-user tabletop spatial data exploration. In *Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 119–122, Seattle, WA, USA. ACM.

Forlines, C., Wigdor, D., Shen, C., and Balakrishnan, R. (2007). Direct-touch vs. mouse input for tabletop displays. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, page 656.

Gallagher, D. (2010). Apple shares jump on iPad sales as rivals emerge - MarketWatch. http://www.marketwatch.com/story/apple-shares-jump-on-ipad-sales-as-rivals-emerge-2010-06-01.

Glassner, A. S. (1993). *Graphics Gems*. Academic Press Inc., 1st edition.

Guiard, Y. and Beaudouin-lafon, M. (2004). Target acquisition in multiscale electronic worlds. *Int. J. Hum.-Comput. Stud*, 61:875–905.

Hayes, S. T., Hooten, E. R., and Adams, J. A. (2010). Multi-touch interaction for tasking robots. In *Proceeding of the 5th ACM/IEEE international conference on Human-robot interaction*, pages 97–98, Osaka, Japan. ACM.

Hjerde, M. (2008). Sender 11: Mobile screen size trends. http://sender11.typepad.com/sender11/2008/04/mobile-screen-s.html.

Hunolstein, S. V. and Zipf, A. (2003). Towards task oriented map-based mobile guides. In *Workshop "HCI with Mobile Guides" at Fifth International Symposium on HCI with Mobile Devices and Services.*

Karat, J., McDonald, J. E., and Anderson, M. (1986). A comparison of menu selection techniques: touch panel, mouse and keyboard. *Int. J. Man-Mach. Stud.*, 25:73–88.

Karlson, A. K. and Bederson, B. B. (2007). ThumbSpace: generalized one-handed input for touchscreen-based mobile devices. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction*, pages 324–338, Rio de Janeiro, Brazil. Springer-Verlag.

Kato, J., Sakamoto, D., Inami, M., and Igarashi, T. (2009). Multi-touch interface for controlling multiple mobile robots. In *Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 3443–3448, Boston, MA, USA. ACM.

Kincaid, J. (2010). Apple has sold 450,000 iPads, 50 million iPhones to date. http://techcrunch.com/2010/04/08/apple-has-sold-450000-ipads-50-million-iphones-to-date/.

Kiriaty, Y. (2009). MultiTouch capabilities in windows 7. http://msdn.microsoft.com/en-us/magazine/ee336016.aspx.

Knott, G. D. (2000). *Interpolating cubic splines*. Springer.

Landay, J. A., L, J. A., and Kaufmann, T. R. (1993). User interface issues in mobile computing.

Lyche, T., Mazure, M., and Schumaker, L. L., editors (2002). *Curve and surface design: Saint-Malo 2002*. Nashboro Press, Brentwood, TN.

Meyer, S., Cohen, O., and Nilsen, E. (1994). Device comparisons for goal-directed drawing tasks. In *Conference companion on Human factors in computing systems*, CHI '94, pages 251–252, New York, NY, USA. ACM.

Micire, M., Desai, M., Courtemanche, A., Tsui, K. M., and Yanco, H. A. (2009a). Analysis of natural gestures for controlling robot teams on multi-touch tabletop surfaces. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 41–48, Banff, Alberta, Canada. ACM.

Micire, M., Drury, J. L., Keyes, B., and Yanco, H. A. (2009b). Multi-touch interaction for robot control. In *Proceedings of the 13th international conference on Intelligent user interfaces*, pages 425–428, Sanibel Island, Florida, USA. ACM.

Microsoft (2010). What is microsoft surface?

Moreno, R. and Mayer, R. E. (2002). Learning science in virtual reality multimedia environments: Role of methods and media. *Journal of Educational Psychology*, 94(3):598–610.

Muratore, D. A. (1987). Human performance aspects of cursor control devices. Technical report, Mitre Corporation.

Nivala, A. and Sarjakoski, L. T. (2003). An approach to intelligent maps: Context awareness. In *Proceedings of The 2nd Workshop on 'HCI in Mobile Guides'*, Udine, Italy.

Oviat, S. L., Coulston, R., and Lunsford, R. (2004). When do we interact multimodally? In *Proceedings of the 6th International Conference on Multimodal Interfaces*, pages 129–136, State College, PA, USA.

Paul, R. (2010). Top 5 most useful apps for artists and designers life scoop. http://mylifescoop.com/featured-stories/2010/10/top-5-apps-for-artists-and-designers.html.

Potter, R. L., Weldon, L. J., and Shneiderman, B. (1988). Improving the accuracy of touch screens: an experimental evaluation of three strategies. In *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*, pages 27–32, Washington, D.C., United States.

Roth, V. and Turner, T. (2009). Bezel swipe: conflict-free scrolling and multiple selection on mobile touch screen devices. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 1523–1526, Boston, MA, USA. ACM.

Roudaut, A., Lecolinet, E., and Guiard, Y. (2009). MicroRolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 927–936, Boston, MA, USA. ACM.

Salomon, D. (2006). *Curves and surfaces for computer graphics*. Birkhuser.

Schöning, J. (2010). Touching the future: The rise of Multi-Touch interfaces. http://www.perada-magazine.eu/view.php?source=002864-2010-03-29.

Seagull, F., Wickens, C. D., and Loeb, R. G. (2001). When is less more attention and workload in auditory, visual, and redundant Patient-Monitoring conditions. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 45:1395–1399.

Shankland, S. (2010). Adobe photoshop for tablets looms nearer | deep tech - CNET news. http://news.cnet.com/8301-30685_3-20021916-264.html.

Smith and Mosier (1986). *Guidelines for Designing User Interface Software*. Natl Technical Information.

Tse, E., Shen, C., Greenberg, S., and Forlines, C. (2006). Enabling interaction with single user applications through speech and gestures on a multi-user tabletop. In *Proceedings of the working conference on Advanced visual interfaces*, pages 336–343, Venezia, Italy. ACM.

Westerman, W. (1999). *Hand Tracking, Finger Identification, and Chordic Manipulation on a Multi-Touch Surface*. PhD thesis, University of Delaware.

Wickens, C. D., Goh, J., Helleberg, J., Horrey, W. J., and Talleur, D. A. (2003). Attentional models of multitask pilot performance using advanced display technology. *Human Factors*, 45(3):360–380. PMID: 14702989.

Wickens, C. D. and Gosney, J. L. (2003). Redundancy, modality, and priority in dual task interference. *Human Factors and Ergonomics Society Annual Meeting Proceedings*, 47:1590–1594.

Wickens, C. D. and Hollands, J. G. (1999). *Engineering Psychology and Human Performance*. Prentice Hall, 3 edition.

Wigdor, D., Perm, G., Ryall, K., Esenther, A., and Chia, S. (2007). Living with a tabletop: Analysis and observations of long term office use of a Multi-Touch table. In *Proceedings of the Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 60–67, Newport, RI.